Universidad
Carlos III de Madrid

Departamento de Ingeniería de Sistemas y Automática

TRABAJO FIN DE GRADO

Generación de trayectorias de caminata para robot minihumanoide de bajo coste

Grado en Tecnologías Industriales

Autor: Roberto Herrera Esteban

Director: Félix Rodríguez Cañadillas
Tutor: Alberto Jardón Huete

Leganés, Septiembre 2014

Título: Generación de trayectorias de caminata para robot minihumanoide de bajo coste
Autor: Roberto Herrera Esteban
Director: Félix Rodríguez Cañadillas
Tutor: Alberto Jardón Huete

EL TRIBUNAL

Presidente: Javier Sanz Feito

Vocal: Francisco José Rodríguez Urbano

Secretario: Plinio Jesús Pinzón Castillo

Realizado el acto de defensa y lectura del Trabajo Fin de Grado el día 30 de Septiembre de 2014 en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de:

VOCAL

SECRETARIO PRESIDENTE

Agradecimientos

A todos mis amigos y compañeros que me han acompañado estos últimos años, en especial a la gente de ASROB que me han ensañado mucho en este tiempo, y a Alazne por acompañarme y haberme motivado en todo momento.

En especial, me gustaría dar las gracias a mis padres y al resto de mi familia por ayudarme a llegar hasta aquí.

Resumen

En este proyecto se pretende desarrollar un sistema hardware/software para la realización de la caminata de un robot mini-humanoide. Este robot es un prototipo de de bajo coste completamente imprimible por una impresora 3D doméstica.

Para ello, se realizará un estudio de los diferentes procedimientos utilizados para esta tarea, se seleccionarán los componentes hardware y se creará el software necesario para la implementación de la tarea de desplazar al robot.

El objetivo final de este proyecto es participar con el prototipo en concursos de robótica mini-humanoide, como CEABOT, en el se pondrán a prueba los desarrollos utilizados.

Palabras clave: Bajo Coste, Mini-Humanoide, Imprimible 3D, CEABOT.

Abstract

This project is intended to develop a hardware / software system for the realization of the walk of a mini humanoid robot. This robot is a prototype of low cost fully printable a home 3D printer.

For this, a study of the different procedures used to perform this task, the hardware components are selected and required to implement the task of moving the robot software is created.

The ultimate goal of this project is to participate in mini-humanoid robotics competitions with the prototype, as CEABOT, in which developments will test used.

Keywords: Low-Cost, Mini-Humanoid, 3D printable, CEABOT.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

En este proyecto se pretende dotar de movilidad a un robot minihumanoide de bajo coste con piezas estructurales totalmente imprimibles, usando para ello una impresora 3D doméstica. Se intentará además usar componentes de bajo coste para dotar de electrónica al prototipo e intentar que no aumente el precio total del robot.

Por otro lado, al ser un robot en una fase de desarrollo baja, al disponer sólo de la estructura y motores que lo componen, es necesario, además de la electrónica, crear todo el *software* necesario para poder controlarlo y, posteriormente, generar trayectorias de caminata que permitan el desplazamiento autónomo del robot.

Al pretender diseñar un robot capaz de tomar decisiones sencillas relativas a su desplazamiento de manera autónoma, es necesario que el nivel de cómputo para las tareas de movimiento realizado a bordo del robot no sea excesivamente alto, más si se considera que la controladora no será tan potente como un ordenador personal. Para disminuir este esfuerzo, se considerará las trayectorias como tareas repetitivas, teniendo que analizar uno de estos ciclos para poder mover al robot correctamente. Además, se intentará procesar previamente las trayectorias intentado ofrecer al robot la respuesta del problema teniendo únicamente que ejecutar las órdenes ya resueltas.

La principal complicación de este proyecto está en no tener datos previos al ser un robot experimental, no pudiendo extrapolar ningún resultado previo obtenido en robots similares. Otra incertidumbre a tener en cuenta son los motores que se utilizan para mover al robot, al no haber sido utilizados en robots similares y presentando características diferentes a estos. Por lo tanto en este proyecto se intentará analizar la viabilidad de estos modelos de motor, indicando si son aptos o no para el movimiento del robot y si son los más adecuados para tal propósito.

Teniendo presente estas ideas, se intentará dar solución a estos problemas a lo largo del proyecto, analizando cada uno de ellos por separado.

1.1. Asociación de robótica ASROB

ASROB, que su logotipo se puede ver en 1.1, es la asociación de robótica de la universidad Carlos III [8] que relaciona a sus miembros, en su mayoría estudiantes de grado y master, con el mundo de la robótica, la automática y la electrónica. Dentro de esta asociación surgió la idea de este proyecto como respuesta a uno de los grupos de trabajo de una de sus áreas.

En la actualidad existen diferentes grupos de trabajo que concentran sus esfuerzos en diferentes proyectos de la rama de la robótica civil. En este momento, existen cinco diferentes agrupaciones que se muestran a continuación.

El grupo de trabajo de los *UAVs* se encargar de diseñar y programar robots aéreos no tripulados orientados a la robótica civil. Sus miembros pretenden diseñar un robot volador

Figura 1.1: Asociación de robótica *ASROB* de la universidad Carlos III.

totalmente autónomo desde el momento de su arranque que sea capaz de tomar decisiones relativas a su posición siguiendo unas premisas dadas por su programador.

Otro grupo de trabajo es el encargado de los *Robots Personales de Competición o RPC*. Las personas de este grupo diseñan y construyen mini robots impulsados con ruedas orientados a competiciones de *sigue-líneas, mini-sumo o robots velocistas* a nivel nacional.

El siguiente grupo de trabajo es el encargado de las *impresoras 3D*. Con la idea de diseñar sus propios robots, la asociación dispone de varias impresoras 3D que permiten tener en pocos minutos las piezas diseñadas por sus miembros. Esto permite crear y replicar pequeños robots con un bajo coste de material.

El *Robot Devastation* es un proyecto que pretende introducir a robots controlados por usuarios en un entorno de realidad aumentada que permite interaccionar unos con otros al estilo de un videojuego *shooter*. En este grupo se programan interfaces que permitan al usuario manejar cualquier tipo de robot móvil de forma remota a tiempo real y poder participar en el juego.

Por último, existe también en la asociación un grupo de trabajo que se encarga de programar *Robots mini humanoides*. Desde su creación, este grupo ha programado y modificado robots comerciales para hacerles participar en competiciones nacionales. Esta asociación trabajó con diferentes tipos de robots mini humanoides que más adelante se detallarán.

Dentro de este último grupo nace el proyecto *MYOD (Make Your Own Droid)* [7] que pretende diseñar un robot mini humanoide totalmente imprimible por un precio inferior a los robot que se comercializan hasta ahora. Esta idea surge como alternativa a poder disponer de un robot mini humanoide por un bajo coste, estando este proyecto orientado a cualquier usuario que pretenda fabricarse su propio robot. Al no se un proyecto destinado sólo para los usuarios de la asociación de robótica, se intentan buscar soluciones comerciales para facilitar el montaje del robot a personas ajenas a la asociación al dar la opción de obtener los elementos necesarios en el mercado o fabricarlos con una impresora 3D doméstica.

El prototipo ha de tener unas características similares al resto de robots comerciales ya que su diseño ha sido orientado hacia la competición nacional de robots mini humanoides *CEABOT* [6] [11]. Tras diseñar el prototipo, nace la necesidad de dotarle de movimiento. Para la competición es necesario que el robot se pueda desplazar por toda la superficie permitida de una forma autónoma. Para poder moverse sobre este plano, el robot ha de poder avanzar, cambiar de dirección y poder desplazarse lateralmente mientras realiza cualquiera de las pruebas de las que se compone el campeonato. De esta necesidad de movilidad surge este proyecto, que intenta dar respuesta a alguno de los problemas que la competición plantea.

1.2. Objetivos

Este proyecto está orientado a generar movimientos que permitan desplazar a un robot de bajo coste por el plano. En el proyecto se verán los diferentes procedimiento que se pueden utilizar para tal propósito y se tratará de buscar soluciones reales al problema.

El prototipo a analizar es el primer robot totalmente imprimible[1] que compite en el concurso nacional *CEABOT*. La característica que le hace único es que todas las piezas estructurales del robot se pueden imprimir con una impresora 3D doméstica pudiendo replicar el robot a un bajo coste de materiales y energía. Además, este robot intenta usar motores analógicos de bajo torque que son mucho más baratos que los que usan sus competidores, abaratando el precio final del robot. Este hecho hace que el precio del robot sea muy inferior a los robot comerciales que participan en la misma competición.

- **Diseñar la electrónica**. En primer lugar, al partir de un estado de desarrollo bajo, el primer paso que ha de darse es la selección de los elementos que componen el sistema eléctrico al mismo tiempo que se define a este. En este punto se pretende alimentar y conectar todos los dispositivos a bordo del robot, como la controladora o los motores. Al querer crear un robot a un precio bajo y que esté abierto a todo el público, es necesario buscar soluciones sencillas, económicas y relativamente fáciles de encontrar en el mercado para poder homogeneizar una solución para cualquier usuario que quiera replicar este modelo.

- **Software**. En segundo lugar, al ser un proyecto con poca trayectoria hasta el momento, no existe ningún tipo de *software* que permita gobernar al robot. Por este motivo es necesario crear librerías y programas que permitan el control del robot así como programas que generen trayectorias para desplazar el robot según unos parámetros que pueda elegir el usuario. También se han de dar las bases de como poder utilizar los programas de un modo concreto para usar correctamente las librerías que en este proyecto se van a diseñar.

- **Optimización del diseño**. En el momento de la redacción de este proyecto, existen dos versiones que configuran la estructura de las pierna, modificando la cadena cinemática del robot. Estas modificaciones se analizarán por separado y se intentará descubrir la viabilidad de ambas y, en el caso de que las dos lo sean, cual de ellas es mejor para generar movimientos en el robot. Por esto, existen fases del proyecto que han de realizarse por duplicado para estudiar las dos configuraciones diferentes, decidiendo finalmente cual de las dos resulta más adecuada.

- **Verificación de la elección de motores y estructura**. Además, este proyecto servirá para probar al robot en situaciones reales de funcionamiento relativas a la movilidad. Por lo tanto se probarán de forma experimental tanto la estructura de la que se parte como de los motores que se emplean. Esto convierte a este proyecto, en cierta medida, en una prueba de viabilidad del robot en cuanto a la fuerza y robustez de los motores así como de la libertad de movimientos que permite la estructura y su entidad para resistir esfuerzos.

- **Homogeneizar datos**. En este proyecto no se utilizarán sensores inerciales ni de fuerza, por lo que no se podrán generar lazos de control para la estabilidad del cuerpo. Por ello es necesario generar movimientos que siempre sean estables en una superficie plana. Por ser un robot que se quiere orientar al público en general, es necesario que dichos movimientos puedan ser usados por diferentes robot del mismo modelo. Por lo tanto, es necesario tener presente en todo momento esta idea, debiendo usar parámetros que hagan homogéneos

[1] Esto se refiere a que la totalidad de las piezas estructurales han sido diseñadas para poder imprimirse en una impresora 3D doméstica de adición de material.

todos los resultados. Aunque todos los robot sean del mismo modelo, pueden existir errores en el montaje que creen diferencias entre un robot y otro, por lo que se ha de crear un método que permita solucionar estas discrepancias y poder usar una misma solución sin tener que modificar esta para adaptarla a uno u otro montaje.

- **Generar una trayectoria de caminata**. El movimiento básico de un robot es el que le permite caminar en línea recta, que será el movimiento que mayor análisis tenga en este proyecto. Para intentar dar una solución real de como afrontar este problema, se utilizarán diferentes métodos intentado adecuarlos a las condiciones de contorno impuestas por el robot y el bajo coste del mismo.
- **Generar otros movimientos**. Para realizar diferentes tareas en es necesario controlar tanto la posición como la orientación del robot. Esto significa que se ha de recurrir a movimientos auxiliares que permitan mover y orientar al prototipo en el plano, adaptándose a las diferentes situaciones a las que el robot puede enfrentarse. Por lo que se indicará como crear diferentes movimientos que permitan desplazar al robot lateralmente y girar sobre sí mismo en ambas direcciones.

Por lo tanto, este proyecto sirve como primer análisis de todos los elementos que en él participan, crear la base de la programación de todos los robot futuros que deseen utilizar una electrónica similar, así como de la centralita y, por último, discriminar que elementos reciclar del diseño para futuras versiones y cuales es necesario descartar.

1.3. Estructura del documento

Para facilitar la lectura de este documento y la comprensión de todas las ideas que aquí se muestran, en este apartado se pretende mostrar la estructura de este proyecto. A continuación, se puede ver a modo de lista un pequeño resumen mostrando el orden en el que aparecen los temas y qué se puede encontrar en ellos.

- En el presente capítulo se muestra una breve introducción de lo que posteriormente se verá. También se muestran los objetivos del proyecto y su origen dentro de la asociación de robótica.
- En el segundo capítulo se explican las características del robot, así como de las restricciones que han marcado su diseño para poder participar en la competición CEABOT.
- El capítulo tercero se presenta el estado actual de los robots que compiten en esa competición, los métodos más extendidos para hacer mover un robot humanoide y además se muestran las herramientas matemáticas que se usan a lo largo del proyecto.
- En el capítulo cuarto se muestra como relacionar las articulaciones del robot con la disposición y orientación de estos en el espacio. En este capítulo se intentan obtener modelos que permitan la movilidad del robot haciendole caminar.
- En el quinto capítulo se habla del diseño del esquema eléctrico del robot así como de los elementos que lo componen.
- El capítulo seis explica las diferentes herramientas informáticas creadas para poder mover el robot, incluyendo programas y librerías.
- El séptimo capítulo experimenta las posibles trayectorias que se usarán para desplazar al robot según los estudios vistos anteriormente. Además se probará cual de las diferentes versiones es mejor y se intentará optimizar el paso del robot. Por último se describirá como realizar otros movimientos del robot.

- En el siguiente capítulo se verán las conclusiones del proyecto, donde se analizarán todos los datos experimentales y se tratará de enfocarlos a un nuevo diseño del robot.
- Finalmente se muestran los costes del proyecto y otros datos similares.

Tras todos los capítulos, aparece una sección de conclusiones donde se anotan los incidentes del proyecto así como las conclusiones que se han podido extraer tanto de la fase de desarrollo como de los experimentos realizados.

Capítulo 2

Marco de trabajo

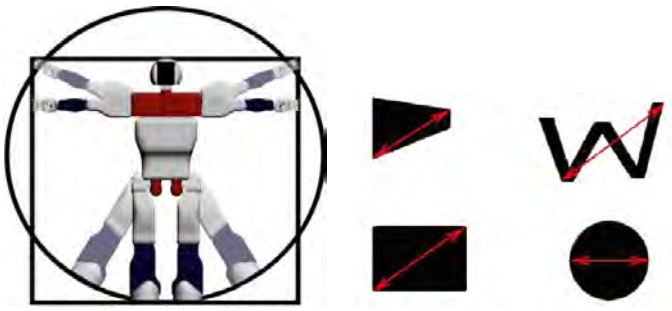
Es este capítulo se pretende dar a conocer el robot diseñado por el grupo de *mini humanoides* de la asociación de robótica. Se pretende mostrar cuales son las dimensiones y características principales del robot, así como de los elementos motrices que lo componen. Además, se mostrará también el concurso al que el robot está destinado, *CEABOT*. Las restricciones de este concurso son las que imponen las condiciones de contorno a este robot, por lo que se explicarán brevemente así como las pruebas que componen al concurso.

2.1. Competición CEABOT

CEABOT [6] es una competición anual que se celebra desde 2006 dentro de las jornadas del *Comité Español de Automática* donde los estudiantes de diferentes universidades compiten con robots mini humanoides en las diferentes pruebas que agrupa. En esta competición se pretende que los estudiantes programen y mejoren los robots comerciales para que dispongan de la autonomía suficiente para superar cada prueba. Para que los robots puedan interactuar con el entorno, es necesario acoplar al robot diferentes sensores que le indique la distancia a un obstáculo cercano o la proximidad de otro robot.

2.1.1. Restricciones

Las restricciones para poder participar en esta competición están limitadas al peso del robot y alguna de las dimensiones características del mismo [3]. Esta competición está destinada a robots con morfología similar a la humana, esto quiere decir que el robot ha de estar formado por un tórax, dos piernas y dos brazos que respeten de manera aproximada las proporciones del *hombre de vitruvio* de Leonaro Da Vinci, como se muestra en 2.1(a). Otra restricción obliga al robot a no tener una altura mayor a los $50cm$. Por motivos de estabilidad, la longitud máxima del pie no puede ser mayor a $11cm$, pudiendo medir esta longitud como se muestra en 2.1(b). La última restricción que han de cumplir los robot es que no han de tener un peso superior a $3Kg$. Además el robot ha de ser totalmente autónomo tanto en alimentación como en procesamiento, no pudiendo ser controlado por el programador una vez arrancado el robot.



(a) Proporciones humanas (b) Máxima longitud del pie

Figura 2.1: Restricciones del diseño.

2.1.2. Pruebas

Este campeonato está compuesto por cuatro pruebas puntuables y una extra a modo de exhibición, cada una destinada a un campo diferente de la robótica mini humanoide.

Carrera de obstáculos

En la primera prueba el robot ha de detectar y esquivar varios obstáculos puestos de forma arbitraria en el campo para intentar llegar al lado apuesto del mismo y regresar al punto de partida. Este campo tiene unas dimensiones de 2x2m estando las líneas de salida y llegada en lados opuestos a 50cm de los laterales, siendo el suelo de color verde y las líneas de color blanco, como se puede ver en la figura 2.2. Los obstáculos tienen unas dimensiones de 20x20x50cm y son también de color blanco. Como se ha dicho, los obstáculos no se colocan de manera predeterminada, por lo que no se puede planificar una ruta. Esto hace que los robots tenga que tener algoritmos para poder detectar y esquivar los obstáculos mediante sensores para poder completar el objetivo final.

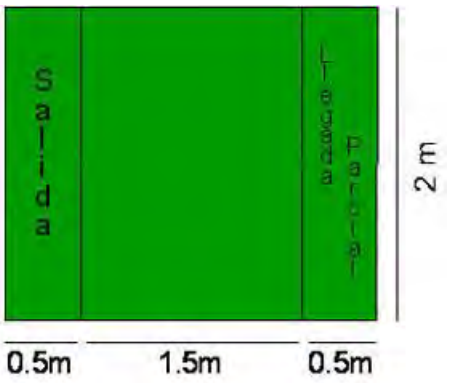


Figura 2.2: Campo de la prueba de obstáculos.

Escalera

Esta prueba de habilidad está basada en el equilibrio del robot y la fuerza de sus motores. Este ha de subir y bajar tres escalones de 3cm de altura colocados a diferentes longitudes unos de otros, tal y como se puede ver en la figura 2.3. En esta prueba se combina el equilibrio del robot con la fuerza de los motores al tener que soportar todo el peso del robot en una sola pierna. Además, para esta prueba es crucial la detección de los escalones y la orientación a la que se llega a estos, que ha de medirse mediante sensores.

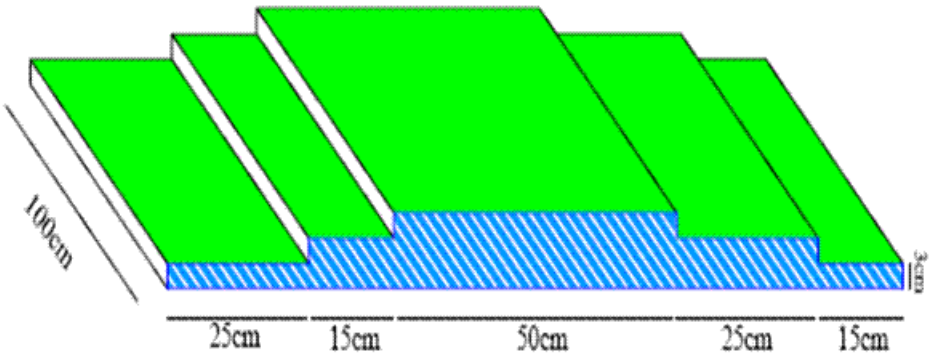


Figura 2.3: Prueba de escaleras.

Lucha

En la prueba de combate los robots han de enfrentarse unos a otro intentando derribarse. Los robots se enfrentan en un *ring* circular de 1,5m de diámetro y son colocados a unos 20cm uno del otro en el centro del mismo, de una forma parecida a como se puede ver en 2.4. Tras una espera de 5s, los robots intentarán derribar al adversario intentando no caerse en el intento. Esta prueba en una combinación de fuerza, estabilidad y velocidad de reacción ante las señales provenientes de los sensores.

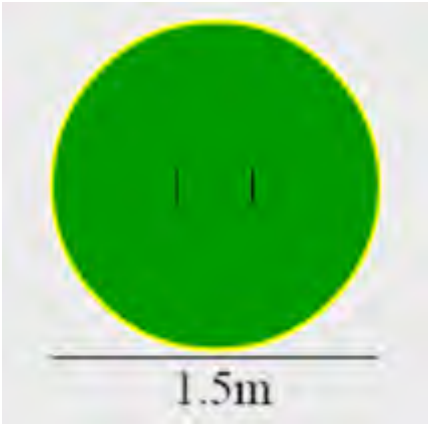
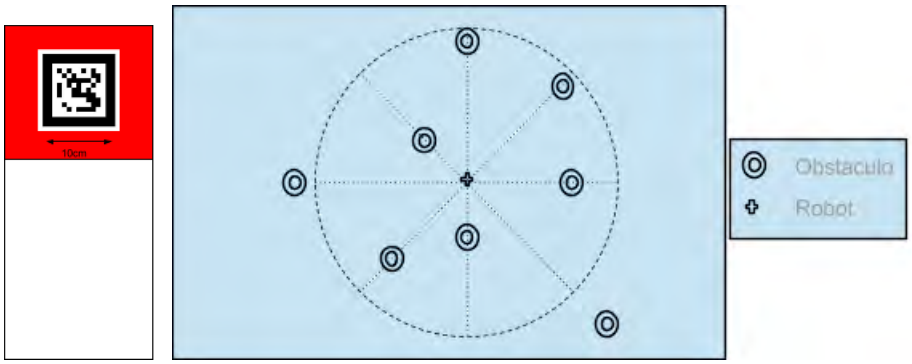


Figura 2.4: Ring para la prueba de sumo.

Visión

En la competición de 2014 se ha incluido una nueva prueba de visión. Los robots han de ser capaces de detectar y desencriptar códigos QR, similares al mostrado en la figura 2.5(a), mediante los cuales se les indica la ubicación del siguiente código, que tendrán que volver a localizar y desencriptar. Esta prueba está destinada a introducir en la robótica mini humanoide de competición un nuevo área, la visión. El sensor principal de esta prueba es una cámara que los robot han de llevar acoplada y el procesamiento de la imagen ha de realizarse a bordo del robot. No obstante, por ser 2014 el primer año que se realiza esta prueba, también está permitido el procesamiento del código de forma externa. Los códigos estarán colocados en la mitad superior de los obstáculos de la primera prueba, separados entre sí 45°. La distancia al centro de estos es variable, pudiendo estar más alejados del robot unos que otros, tal y como se ve en 2.5(b).



(a) ejemplo de código QR

(b) Disposición de los códigos

Figura 2.5: Prueba de visión.

Prueba libre

La competición permite una prueba adicional donde se puede mostrar otras habilidades extras que pueda tener el robot. Esta prueba no presenta ningún tipo de restricción ni puntuación para la clasificación final y está orientada a la danza o cualquier otra habilidad programada en el robot.

2.2. Robot MYOD

El actor principal de este trabajo es el prototipo desarrollado por el grupo de trabajo MYOD dentro de la asociación de robótica ASROB de la universidad Carlos III [7], mostrado en la figura 2.6. Este prototipo es del tipo Mini-Humanoide, siguiendo unas proporciones antropomórficas pero a una escala menor e intentándose ceñir a la normativa del campeonato *CEABOT* antes explicado.

Con la idea de acercar la robótica minihumanoide al público en general, este prototipo ha sido diseñado para que todas las piezas estructurales se puedan imprimir con una impresora 3D doméstica. El resto de piezas, como la tornillería o la electrónica, se pueden conseguir de manera sencilla en ferreterías o tiendas especializadas. Esto permite que el robot se pueda replicar por cualquier persona interesada con un bajo coste.

Para concretar más, todos los estudios se realizarán sobre la primera versión estable de este robot, incluyendo algunas de las primeras variaciones de la estructura. Para familiarizar al lector con este prototipo, se detallará cada uno de los elementos partícipes en este robot así como alguna de sus características principales.

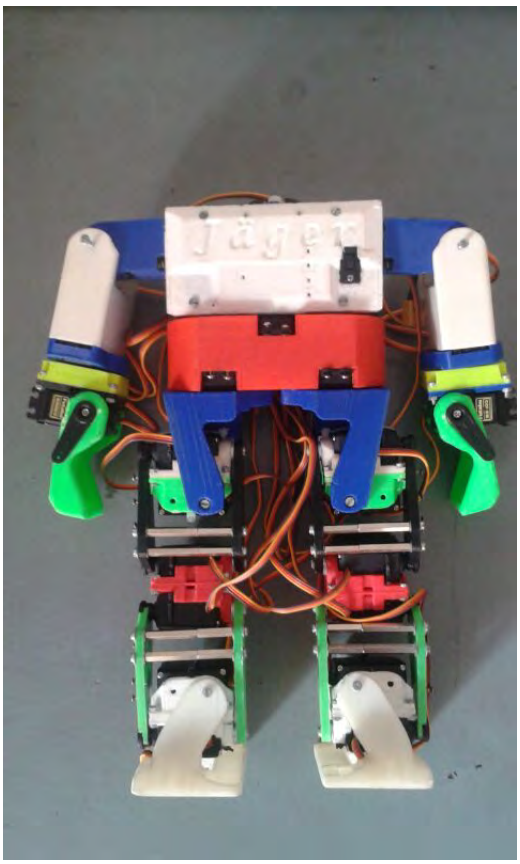


Figura 2.6: Robot MYOD.

2.2.1. Descripción General

El prototipo MYOD dispone de 24 grados de libertad, 7 en cada pierna, 4 en cada brazo, 1 haciendo las veces de cintura y 1 en la parte superior del cuerpo a modo de cuello.

En posición erguida el robot tiene una altura de 33 centímetros, sin incluir la cabeza, y un peso aproximado de 2,35 kilogramos[1], incluyendo en este cálculo el peso total de la estructura, los motores, la electrónica básica y la batería.

En este estudio se van a usar las dos primeras versiones de la estructura de las piernas, que presentan diferencias entre sí. Dependiendo de la versión de del robot las dimensiones varían en función de los cambios realizados. El cambio más notable se puede ver en la pieza que une la cadera con la pierna. Como se ve en la figura 2.7, la distancia entre los ejes que unen estas piezas con el resto de la pierna varía en función de la versión, siendo $106mm$ en la primera y $81mm$ en la segunda. Además, con el cambio realizado en la segunda variación, los ejes perpendiculares al plano frontal quedan alineados verticalmente. Otro cambio que produce esta variación es que eleva la altura total del robot $12mm$.

[1]Esta medición ha sido realizada para la configuración de cadera estrecha, 23 motores TowerPro MG996r, un TowerPro MG90s, Arduino Mega 2560 con shield, regulador UBEC y LiPo de tres células con una capacidad de 2200ma·h

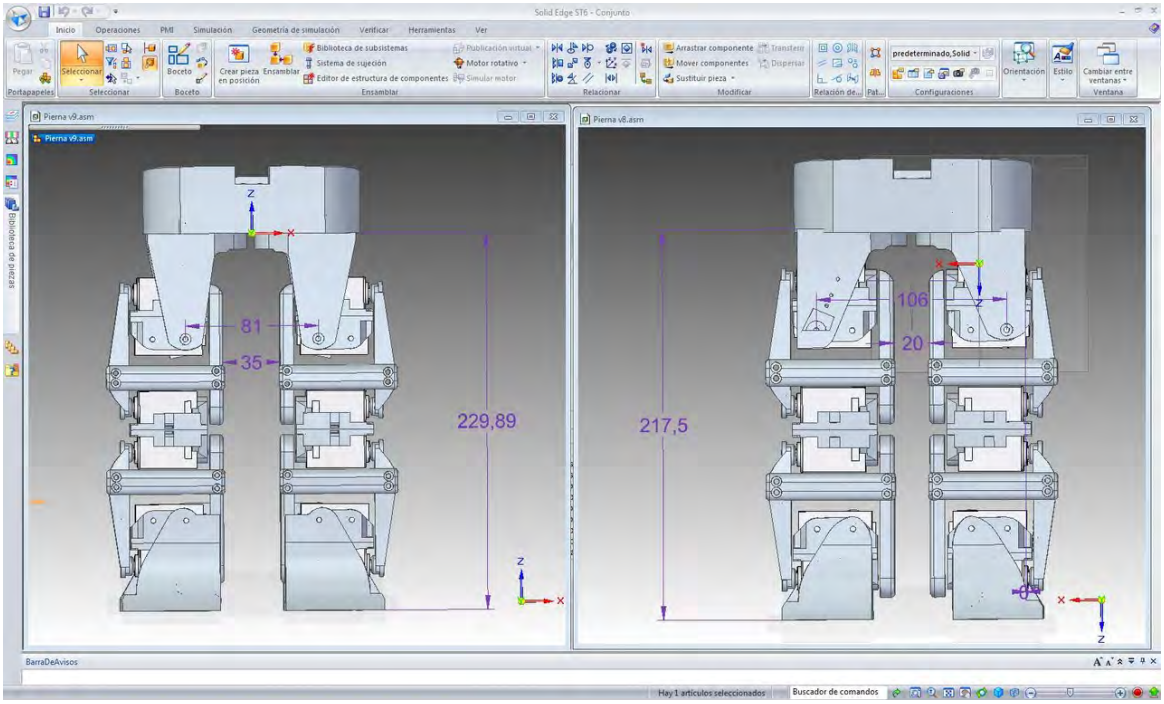


Figura 2.7: Diferencias entre las dos versiones del robot.

Descripción de la pierna

Cada una de las dos piernas dispone de siete grados de libertad y siete eslabones. Empezando desde la cadera del robot hasta el extremo de la pierna, se numerará a continuación cada uno de los grados de libertad y eslabones de esta cadena cinemática [2]. Por ser simétricas, sólo se detallará una de las piernas, numerando la otra de forma similar. Es importante tener en cuenta esta numeración ya que a lo largo de este proyecto se recurrirá a ella frecuentemente.

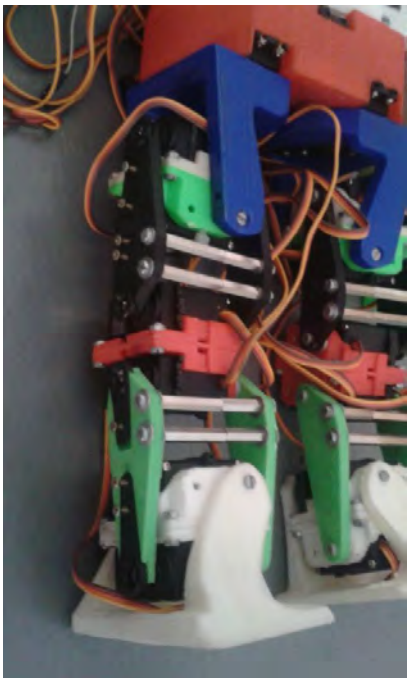


Figura 2.8: Pierna del Robot MYOD.

Para numerar cada uno de los eslabones y ejes de cada pierna, se toma como base la cadera

[2]Se seguirá el convenio de *DH* para numerar los eslabones y articulaciones [23].

del robot. Desde este momento la cadera será considerada como la *base o eslabón cero* de cada una de las piernas así como el sistema de referencia fijo.

El primer grado de libertad está situado sobre la cara inferior de la cadera, apuntando hacia el suelo, permitiendo orientar el conjunto de la pierna rotando el primer eslabón sobre el eje vertical. Los dos siguientes grados de libertad están situados en la parte superior de la pierna, en el extremo del primer eslabón. Estos dos ejes que se cruzan, junto al antes mencionado, permiten emular el movimiento de la ingle del robot. El soporte que une estos dos ejes que se cortan en el espacio configura el segundo eslabón de la extremidad. A la salida del tercer eje están ubicadas dos piezas que imitan la parte superior de una pierna antropomórfica, y forman el tercer eslabón.

En el extremo del tercer eslabón se encuentran los dos siguientes ejes que ambos forman la rodilla de este prototipo. Estos ejes son paralelos entre sí y, del mismo modo que lo mencionado antes con los ejes dos y tres, el soporte de ambos configura el cuarto eslabón. Al continuar descendiendo por la extremidad, en el extremo del quinto eje aparecen de nuevo dos piezas estructurales que imitan la parte inferior de una pierna humana y son consideradas como quinto eslabón. Al otro lado de este eslabón se encuentran los dos últimos grados de libertad, estos dos ejes que se cruzan emulan el tobillo del robot, considerando como el sexto eslabón a la unión entre ambos y, por último, el pie como séptimo y último eslabón.

Todo lo descrito hasta ahora, se puede ver de forma gráfica en la figura 2.9.

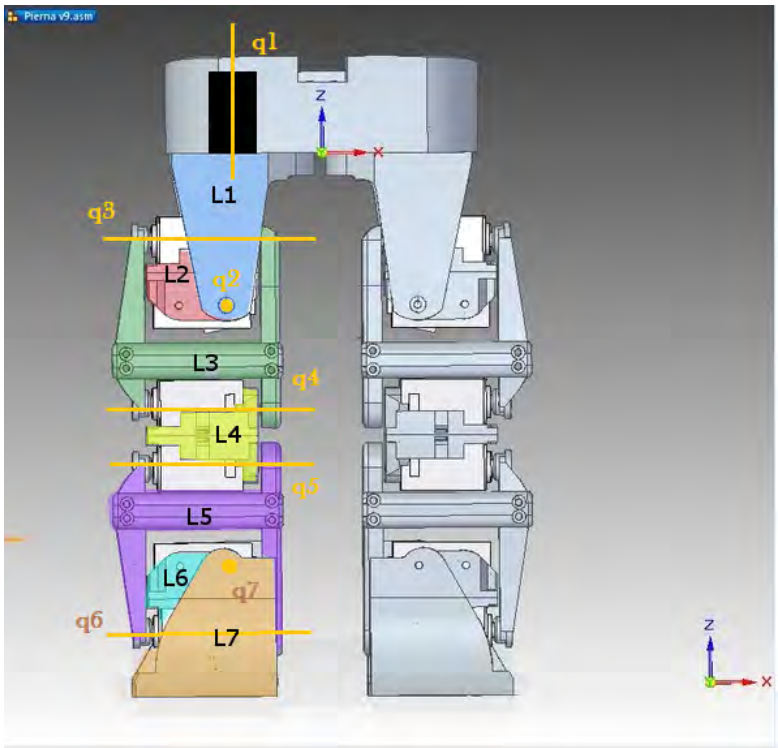


Figura 2.9: Definición y numeración de los elementos de la pierna.

Descripción del brazo

Los dos brazos del robot tienen 4 grados de libertad cada uno. De modo similar a las piernas, el pecho del robot se considera como base o eslabón cero para cada uno de estos dos conjuntos. El primer grado de libertad aparece perpendicular a una de las caras laterales del pecho del robot, y solidario a él está el primer eslabón del brazo. Al extremo de este se encuentra el segundo eje del conjunto. Estos dos ejes que se cortan emulan una articulación esférica que hace las veces del hombro del robot. A la salida de este eje se encuentra el segundo eslabón del

conjunto. Paralelo a la generatriz del eslabón anterior se encuentra el tercer grado de libertad del brazo, que permitirá cambiar la orientación del tercer eslabón. Apoyado sobre dicho eslabón se encuentra el cuarto y último grado de libertad, que se cruza con el anterior. Por último, el cuarto eslabón se encuentra en el extremo del conjunto, imitando la forma de una manopla o pinza dependiendo de la versión de la última pieza. Se puede ver lo mencionado en en la figura 2.10.

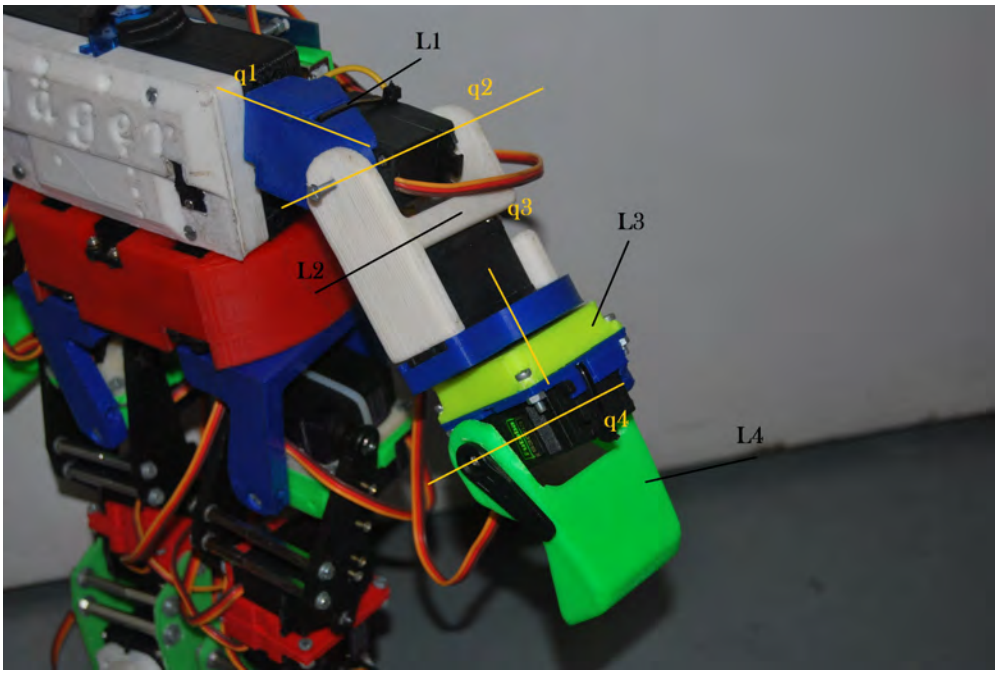


Figura 2.10: Brazo del Robot MYOD.

Descripción del cuello y la cintura

Los dos últimos grados de libertad que configuran al robot se encuentran en la parte inferior y superior del pecho. El primero de ellos es el eje correspondiente a la cintura.

Este eje es normal al plano horizontal y tiene su eje de acción en la parte inferior del pecho. Sobre dicho eje se produce el giro relativo entre la cadera y el pecho, permitiendo que el robot tenga orientaciones diferentes para el conjunto de las dos piernas y el pecho.

De modo similar, el último eje se encuentra en la parte superior del pecho y es perpendicular al plano horizontal. Este eje es el encargado de producir un giro relativo entre el torso del robot y la cabeza de este, pudiendo orientarla de forma independiente.

2.2.2. Motricidad del prototipo y servomotores

El prototipo ha sido diseñado teniendo en cuenta los diferentes tipos de servomotores para poder mover cada articulación y así poder desplazar al robot por el plano o generar otro tipo de movimientos. A continuación se explica el funcionamiento básico de uno de estos servomotores y, posteriormente, los modelos de servomotor que han sido seleccionados para tal propósito.

Funcionamiento y principios básicos

Para dar movilidad al robot se han usado servomotores para producir el giro relativo entre los diferentes elementos. Los servomotores son elementos motrices que posicionan su piñón de salida en el ángulo requerido. Dicho objetivo se consigue con un motor de corriente continua, un tren de engranajes reductores, un *puente H* que permite que el motor gire en ambos sentidos

y los elementos de control de posición, como el *encoder* y el comparador, elementos que se ven en la figura 2.11.

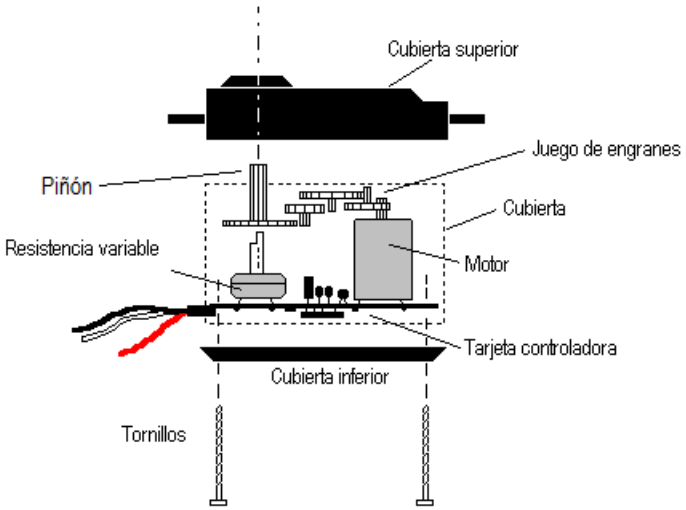
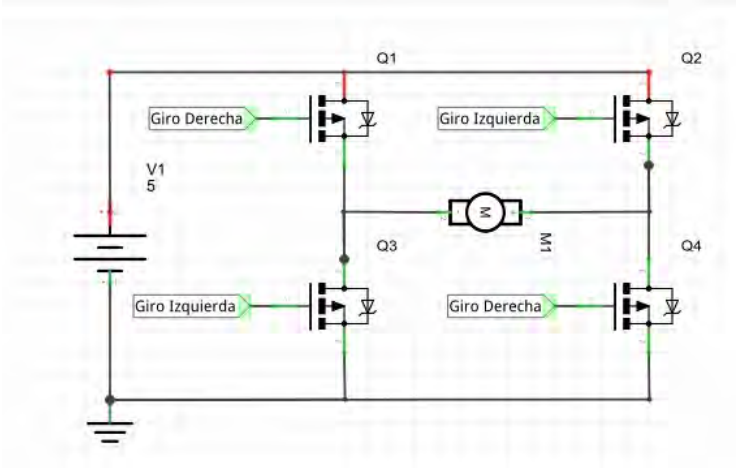
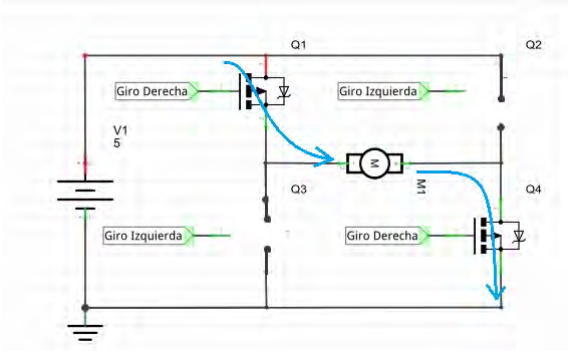


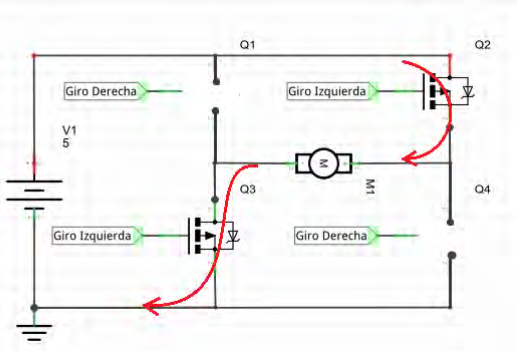
Figura 2.11: Elementos de un servomotor.



(a) Esquema eléctrico del puente H.



(b) Giro hacia la derecha.



(c) Giro hacia la izquierda.

Figura 2.12: Cambio de giro de un motor por la acción del puente H.

El piñón de salida del servomotor tiene solidario a él un *encoder* que indica en qué posición se encuentra este último engranaje. Mediante un circuito de control, se compara la posición del piñón con la posición que se requiere, indicada a través de la señal de entrada. En función

del resultado de esta comparación, el circuito de control selecciona que par de transistores del *puente H* son los que se deben activar para que se produzca el giro del motor en un sentido u otro. Esta comparación intenta hacer que la diferencia entre la posición actual y la deseada sea cero, por lo que para tratar de reducir este valor se ha de girar el motor de continua en el sentido de giro correcto para cada caso, como se ve en 2.12. Con todo esto, se consigue posicionar el piñón en la posición deseada.

Para este prototipo se ha tenido en cuenta las líneas de pensamiento de bajo coste y la accesibilidad al público en general de cualquier material, por lo que el grupo de trabajo MYOD ha seleccionado los siguiente modelos de servomotores para producir el movimiento de forma controlada de todos los elementos. Pese a no ser los modelos más adecuados para dar movilidad a un robot humanoide a escala, todos cumplen las características mínimas para poder mover fluidamente cada articulación, ciñéndose a las corrientes de pensamiento ya citadas. En concreto se han elegido diferentes modelos de servomotores en función del lugar donde van alojados y de la tarea correspondiente, optimizando el peso, el consumo y las necesidades requeridas de torque.

Motores usados en el prototipo

TowerPro MG996R Este motor se ha elegido específicamente pensando en las piernas y la cadera del robot. Estos servomotores tienen en su interior engranajes metálicos que les permiten soportar mayores esfuerzos sin deteriorar la parte mecánica. La elección de estos servomotores para las articulaciones especificadas se debe a que esas uniones son las que más tensiones sufren al sustentar la mayor parte del peso del robot, a la vez que sufren más esfuerzos dinámicos en el movimiento de caminata. En la tabla 2.1 se pueden ver las especificaciones[3] de este tipo motor [20].

Control	Analógico (PWM)
Alimentación	4.8-7V Recom: 6V
Rango de rotación	0-180º
Torque Máx.	10Kg·cm
Velocidad de giro	20ms/60º
Sentido de giro	horario
Peso	55g
Dimensiones	40.6x19.8x42.9mm
Engranajes	Metálicos
Conector	JR
Señal de control	20ms Periodo/min 10ms max 20ms
Consumo aprox. en reposo	100mA
Consumo transitorio(pico)	800mA

Tabla 2.1: Características del servo TowerPro MG996R.

El precio aproximado de estos motores ronda los 8.50 €[12].

[3]Los valores de consumo energético se han obtenido experimentalmente para cada tipo de motor. El valor de consumo transitorio ha sido obtenido en unas condiciones en las que el motor no presentan carga de trabajo.

Figura 2.13: TowerPro MG996R.

Futaba S3003 Este tipo de motor se utiliza específicamente para las articulaciones de los brazos. El motivo para ello es ejercen menos par que los *TowerPro MG99R*, tiene un menor peso con unas dimensiones similares y, además, el consumo energético es menor. Al tener los engranajes de plástico en vez de metálicos, se reduce el peso total del robot sin perder movilidad en ningún eje. Por presentar unas dimensiones similares a los *TowerPro MG996R* se pueden sustituir fácilmente unos por otros en función de las necesidades de potencia y par en diferentes situaciones. Al necesitar menor fuerza en las extremidades superiores del robot, el grupo de trabajo MYOD ha optado por implantar este tipo de motor en todas las articulaciones de los brazos. En la tabla 2.2 se especifican todas las características de este modelo de servomotor [20].

Control	Analógico (PWM)
Alimentación	4.8-7V Recom: 6V
Rango de rotación	0-180º
Torque Máx.	4Kg·cm
Velocidad de giro	20ms/60º
Sentido de giro	Antihorario
Peso	37g
Dimensiones	39.9x20.1x36.1mm
Engranajes	Plástico
Conector	JR
Señal de control	20ms Periodo/min 10ms max 20ms
Consumo aprox. en reposo	100mA
Consumo transitorio(pico)	600mA

Tabla 2.2: Características del servo Futaba S3003.

En la actualidad se se puede encontrar este motor por un precio en cercano a los 8 €.

Figura 2.14: Futaba S3003.

TowerPro MG90S Este tipo de *micro servo* presenta unas dimensiones muy reducidas, un consumo muy bajo, bajo par y muy poco peso, por lo que le hace idóneo para hacer las veces del eje del cuello del prototipo. A continuación, en la tabla 2.3, se muestras las características de este motor. [20]

Control	Analógico (PWM)
Alimentación	4.5-6V Recom: 5V
Rango de rotación	10-170º
Torque Máx.	2.4Kg·cm
Velocidad de giro	15ms/60º
Sentido de giro	antihorario
Peso	14g
Dimensiones	23.1x12.2x29.0mm
Engranajes	Plástico
Conector	JR
Señal de control	20ms Periodo/min 10ms max 20ms
Consumo aprox. en reposo	25mA
Consumo transitorio(pico)	200mA

Tabla 2.3: Características del servo TowerPro MG90S.

Esto motor tiene un coste aproximado de 4 €[12].

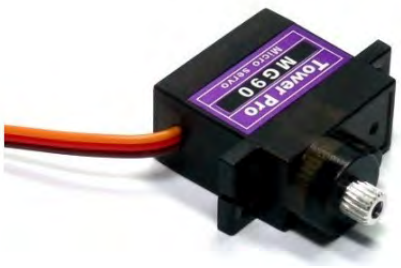


Figura 2.15: TowerPro MG90S.

Tras ver los diferentes motores empleados por el prototipo, se pueden comparar con otros montados en robots que participan en *CEABOT*. También se puede compara el precio de estos y las estructuras de otros robots comerciales.

Capítulo 3

Estado del arte

En este capítulo se pretende mostrar el estado actual de la robótica humanoide. Se intentará explicar las líneas de desarrollo presentes en los últimos años en este campo de la robótica y que objetivos se pretenden alcanzar al medio plazo. Además se mostrarán unos ejemplos de diferentes robots comerciales que pueden participar en la competición *CEABOT*. Por otro lado, se verán el estado del arte en cuanto a caminatas bípedas orientadas a la robótica y, por último, se mostrarán las diferentes herramientas matemáticas orientadas a brazos robóticos que se utilizarán en el proyecto.

3.1. Introducción

En la actualidad existen diferentes tipos de robots humanoides, tanto a tamaño real como a pequeña escala. La robótica en los últimos años ha tenido avances muy relevantes en el campo de los humanoides. Anualemente se celebra la competición *Darpa Robot Chalenger o DRP* [5] donde se muestran los últimos avances realizados en la robótica a nivel internacional, especialmente en robots móviles, tales como los que aparecen en 3.1. Los robots bípedos tienen un lugar reservado en esta competición donde cada año aparecen nuevos avances en la estabilidad y movilidad de robots antropomórficos, capaces de resolver diferentes problemas de movilidad al relacionarse con el entorno de una forma similar a la humana.

Por otro lado, los robots antropomórficos se consideran parte de un futuro a medio plazo, ya que pueden desplazarse y reaccionar con un entorno diseñado para humanos, sin tener que modificarlo para adaptarlo a la máquina. Algunas empresas han tomado esta dirección al considerar a la robótica móvil, y aun más la humanoide, como un negocio emergente, destinando esfuerzos y recursos en las áreas de I+D+I destinadas a la robótica.

Figura 3.2: Robot impulsado con ruedas.

El principal inconveniente de los robots bípedos es la complejidad de movimiento si se les compara con robots móviles impulsados por ruedas o que se pueden desplazar por circuitos

(a) Evolución del robot ASIMO de Honda.

(b) Big Dog de Boston Dinamic.

(c) Army's Battlefield Extraction Assist Robot (BEAR).

(d) Robot humanoide de Boston Dinamic.

Figura 3.1: Estado del arte en la robótica móvil.

cerrados, como raíles o vías, como el robot de la figura 3.2. Para mover a estos robots bípedos es necesario tener en cuenta los efectos dinámicos que aparecen en el robot durante el movimiento del mismo, por lo que en muchas ocasiones es necesario disponer de un control de estabilidad si se quiere introducir en un ambiente real con una superficie de apoyo irregular. Junto con la complejidad del control cinemático de cada pierna, se añade el problema de la estabilidad, que han de resolverse conjuntamente y a tiempo real si no se quiere que el robot se desequilibre y caiga.

Los robots humanoides están destinados a suplir al hombre en tareas peligrosas para evitar daños personales. Algunas líneas de pensamiento están orientadas a esto, diseñando robots a escala similar a la humana que permitan desenvolverse en entornos de riesgo, como catástrofes o ambientes tóxicos, de forma autónoma o tele-operada. La elección de los robot humanoides para este fin suele estar condicionada por el difícil acceso y la necesidad de interaccionar con el entorno de la misma forma que lo haría una persona, al estar muchos de los sistemas o mecanismos orientados a la manipulación por humanos.

También se pretende introducir el mundo de la robótica en los hogares, programando a robots para que se encarguen de alguna de las tareas domésticas del día a día. Uno de los robots más extendidos en la actualidad es el robot *Roomba*(figura 3.3), empleado para la limpieza de suelos. Se prevé que estos pequeños robots sean cada día más comunes, pudiendo realizar diferentes tareas repartiéndolas entre varios robots. También existe una previsión a largo plazo de que estos robots serán sustituidos por humanoides que podrán realizar viarias tareas domésticas.

Figura 3.3: Robot de limpieza Roomba.

Por otro lado, existen robots humanoides con menores dimensiones, aunque con una complejidad similar. Estos robots suelen estar destinados al entorno académico y de investigación, intentando solucionar problemas similares pero en un entorno adaptado a sus dimensiones. Los más comunes se pueden ver en la figura 3.4.

Figura 3.4: Robots humanoides para uso educacional.

3.2. Robot Comerciales que pueden participar en CEABOT

Como se ha dicho, el robot *MYOD* ha sido diseñado con unas dimensiones y características específicas para poder participar en la competición *CEABOT*. En esta competición suelen participar cada año diferentes universidades llevando robots comerciales que son puestos a punto y modificados por estudiantes de grado y master. A continuación se muestran los diferentes robots comerciales, así junto a sus características y prestaciones, que cumplen las restricciones de la competición.

3.2.1. Robonova-1

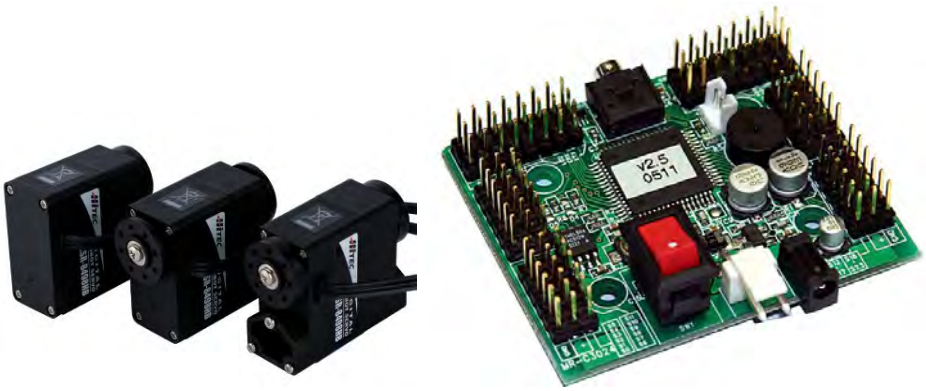
Este robot de *Hitec* tiene una altura aproximada de 30,5cm en posición totalmente estirada y un peso de 1,3Kg [21] [1]. El modelo *Robonova-1* dispones de 16 grados de libertad, 5 en cada pierna y 3 en los brazos. No dispone de movilidad en la cabeza ni cintura, ni tampoco puede variar la orientación de la pierna con respecto a la cadera.

[1] Este modelo ha dejado de ser comercializado por *HITEC*, por lo que es necesario recurrir a terceros para obtener información actualizada.

Figura 3.5: Roobonova-1.

Los motores HSR 8498HB son los encargados de mover las articulaciones del robot. Estos elementos disponen de $7{,}4Kg_f \cdot cm$. Los servomotores son controlados mediante una señal PWM pero permiten leer mediante otra comunicación la posición y el consumo de tensión y corriente de ese motor. Este sistema de comunicación permite leer la posición real de cada articulación, permitiendo capturar la posición real del robot para utilizarla posteriormente en futuros movimientos. Este sistema de captura de movimientos es llamado *Catch & Play* y permite programar la movilidad de robot de una manera muy sencilla, moviendo el robot como un maniquí o muñeco articulado y guardando las posiciones con un ordenador.

La controladora de este robot está basada en el microcontrolador *Atmel ATMega 128*, que dispone de 40 GPIOs(*General Purpose Input/Output* y la posibilidad de comunicación con ella mediante I^2C o comunicación serie.



(a) Motor HSR 8498HB (b) Controladora

Figura 3.6: Principales elementos de Robonova.

Este robot ha de ser programado en el lenguaje *RoboBasic*, que es un tipo de lenguaje basado en *Basic* pero adaptado para la movilidad del robot. Además, este kit incluye otros programas y complementos que permiten la programación en un entorno gráfico.

La alimentación de este robot se realiza mediante una batería de níquel de $6V$.

El precio de cada uno de los motores ronda los 43,65 €, pudiendo adquirir el kit completo por 934,05 €[21].

3.2.2. Bioloid

El kit que ofrece *ROBOTIS* dispone de numerosas piezas y motores que permiten combinarlos de diferentes formas pudiendo montar un robot hexápodo, un cuadrúpedo, un robot similar a un pequeño dinosaurio y varias formas humanoides [19]. En concreto se hablará del modelo humanoide *Type A* que presenta 18 grados de libertad y es el más empleado en la competición. En esta versión, el robot pesa 1,7Kg y tiene una altura de 39cm con cabeza.

Figura 3.7: Bioloid Type A.

Este robot está formado por 6 articulaciones en cada pierna y 3 en cada brazo. Esto hace la suma total de todos los grados de libertad disponibles en el robot, por lo que no presenta movilidad en la cintura ni en el cuello. Para dar movimiento a estas articulaciones se utilizan motores *Dynamixel AX-12+* con un toque de $14Kg \cdot cm$ y permiten trabajar en rotación continua, por lo que no existe un tope mecánico que limite el rango del movimiento. Estos motores son controlados mediante un protocolo *half-duplex* similar al I^2C que permite conectar a todos los motores en *daisy chain*. Esta conexión de motor a motor permite ahorrar en el número de cables que existen en el robot. Estos motores permiten hacer lecturas de posición, torque, intensidad y tensión al mismo tiempo que tienen registros en los que se pueden guardar instrucciones de velocidad angular, posición o toque que se puedan usar en el siguiente movimiento. Para acceder a estos motores o posicionarlos, al estar conectados en cadena, es necesario saber la etiqueta numérica o dirección que corresponde a cada uno, enviándoles datos de ese modo.

(a) Controladora (b) Dynamixel AX-12+ conectados en Daisy Chain

Figura 3.8: Principales elementos del Bioloid.

La controladora de este robot es la *CM-510*, aunque existen otros modelos que ofrece *ROBOTIS* para tal fin. Esta controladora dispone de 6 puertos para conectar sensores y otros 5 puertos que permiten la conexión de los motores.

El entorno de programación de estos robot es bastante cerrado, al poder programarlo en un lenguaje de bloques que sólo permite cambiar ciertos parámetros. Esto ofrece poca libertad al programador. Además, la propia empresa vende sus sensores adecuados para el robot, existiendo una pequeña gama de productos que se pueden adaptar el kit, aunque vengan diseñados y orientados al robot. Con estos motores también se puede utilizar *Catch & Play* para capturar movimientos, además de disponer de un entorno gráfico que permite ver como se mueve el robot según los movimientos y posiciones capturados o programados.

Por último, este robot es alimentado por una batería con un voltaje nominal de $11.1V$.

El precio de mercado de este kit ronda los 985 €y se pueden adquirir los motores por 43 €[17] cada uno.

3.2.3. Kondo KHR-3

El robot de la empresa japonesa *Kondo* tiene 17 grados de libertad, en la mismas disposición que el *Bioloid* y añadiendo uno a la cintura. La misma empresa dispone de otros modelos de robots humanoides pero no se detallarán. El modelo *KHR-3* tiene una altura en posición erguida de $34cm$ y un peso de $2{,}7Kg$ [4].

Figura 3.9: Kondo KHR-3.

Este robot está impulsado por los motores *KRS-2552HV ICS* con $14Kg \cdot cm$ alimentados a 11.1V con una batería de niquel. Estos servos también están preparados para conectarse en *daisy chain* y presentan características similares a las detallas para los servomotores del *Bioloid*.

El procesamiento de movimientos y sensores se realiza a través de la controladora *RCB-4*.

(a) Controladora RCB-4. (b) KRS-2552HV ICS.

Figura 3.10: Principales elementos del Kondo KHR-3.

Este robot se puede adquirir por un precio de 1,829 $ y cada uno de sus motores por 110 $.

3.3. Estado del arte en caminatas humanoides

En la actualidad existen diferentes métodos que permiten la movilidad de robots bípedos. Uno de los más utilizados es el *Zero Momento Point o ZMP*. En este método se pretenden dos cosas, que el punto de centro de presiones este siempre proyectado verticalmente en una región estable, tanto para la fase de apoyo simple como para la de apoyo doble, y que los esfuerzos resultantes de la inercia y de las fuerzas gravitatorias sean cero.

El concepto de ZMP fue introducido en la robótica por Miomir Vukobratovic y sus compañeros [15] [22] [14]. En él se supone que la reacción del robot contra el suelo en la fase de apoyo simple aparece en la suela de dicho pie del robot. Estos esfuerzos distribuidos se han de agrupar en un punto P que represente todos los esfuerzos R presentes, como se ve en la figura 3.11. Por otro lado, para evitar tener en cuenta la compleja dinámica presente en la parte superior del cuerpo, se puede considerar únicamente las reacciones que parecen en la articulación del tobillo A, simplificando los cálculos posteriores.

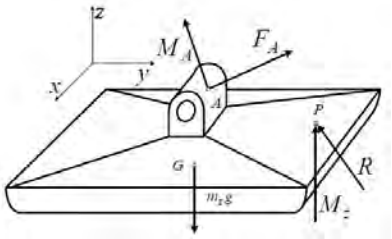


Figura 3.11: Fuerzas y momentos presentes en el pie de apoyo.

Si se descomponen tanto las fuerzas resultantes como los momentos (R_y,R_y,R_z,,M_x,M_yyM_z), y además se considera el pie inamovible al existir rozamiento entre el pie y el suelo, se puede ver que se compensan las reacciones horizontales del tobillo(F_x y F_y) con las del punto de apoyo. Siguiendo el mismo razonamiento, se han de compensar los momentos verticales (M_z), que de no ser así harían rotar el pie del robot sobre el plano.

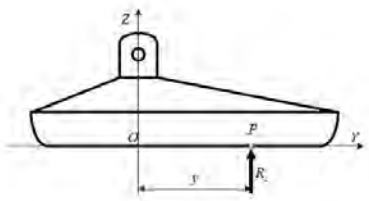


Figura 3.12: Compensación de los momentos con el punto P. ZMP.

Por otro lado, en el instante de equilibrar los momentos horizontales(Ma_x y Ma_y) que aparecen en la unión del pie con el resto del cuerpo, se ve que es necesario desplazar el punto de aplicación de las reacciones verticales R_z para que este sea capaz de oponerse a él, como muestra la figura 3.12. Este desplazamiento ha de realizarse dentro de la huella del pie sin que este salga fuera, ya que no existiría la reacción como tal. Por lo tanto, un aumento en los momentos laterales del tobillo se han de compensar desplazando el punto de reacción. Al hacer que las reacciones en los momentos sean nulas, es lógico llamar a este punto *Zero Momento Point o ZMP.*

Si en algún momento el ZMP sale de la huella del pie durante la fase de apoyo simple en un pie, se puede considerar que se ha iniciado el vuelco del robot, al rotar sobre el borde del pie en lugar de sobre el punto inmaterial que indican los cálculos.

$$\vec{R} + \vec{F_a} + m\vec{g_s} = 0$$

$$\vec{OP}\mathrm{x}\vec{R} + \vec{OG}\mathrm{x}m_s\vec{g} + \vec{M}_A + \vec{M}_z + \vec{OA}\mathrm{x}\vec{F}_A = 0$$

Por lo tanto, en este estudio se buscan movimientos que las reacciones que produzcan en el pie del robot siempre estén dentro de la huella del mismo. Para no considerar todos los esfuerzos dinámicos del robot, se pueden poner sensores que midan el torque que se produce en las articulaciones del tobillo, dando una idea muy aproximada de los esfuerzos reales que aparecen en el robot en lugar de tener que realizar un análisis dinámico detallado.

Partiendo de esos datos de torque, se puede realizar un lazo de control que haga que los movimientos de la parte superior del robot que generan las reacciones en el tobillo creen pares menores a los que el robot puede oponerse en la reacción del pie o que la proyección punto de presiones del robot se proyecte en el ZMP.

Una variante de este método es el llamado *Cart - Table.* A la hora de realizar el control de los esfuerzos dinámicos que se producen en el robot, se puede considerar como un plano horizontal a la altura del centro de gravedad por el cual se desplaza un carro que representa la masa total del robot, como se ve en la figura 3.13. Este plano horizontal ha de considerarse sujeto al punto de reacción P mediante una barra ideal indeformable capaz de transmitir los esfuerzos. Para generar los esfuerzos dinámicos necesarios, se puede considerar que estos son provocados por los desplazamientos del carro. Es una forma de modelado simple y sencilla, que junto a una cinemática que relacione los esfuerzos modelizados por el carro con las articulaciones reales del robot se consiguen los efectos deseados.

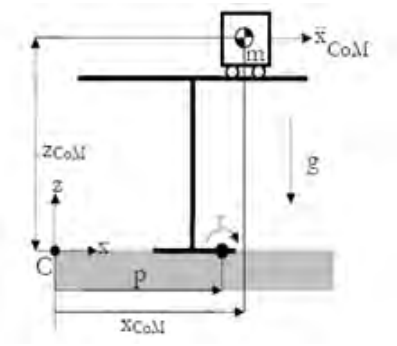


Figura 3.13: Modelado de Cart Table.

3.4. Comparativa de placas de control

En el mercado existen diferentes elementos que se pueden usar como controladoras para robots. En los siguientes apartados se pretenden mostrar algunos *microcontroladores* y *single board computer o SBC* o también *microordenadores*.

3.4.1. Arduino Mega 2560

La primera controladora de la que se hablará será la placa *Arduino Mega 2560*, basada en el microcontrolador de *ATMEL atmega2560*[1] con arquitectura AVR. Esta placa no es más que un microcontrolador reprogramable que permite realizar tareas de forma repetitiva mediante el uso de salidas y entradas digitales. Este tipo de recurso está más orientado a proyectos con bajo nivel de cómputo y teniendo vinculados a la placa sensores y actuadores con comunicaciones de bajo nivel.

Este modelo de *Arduino* dispone de 54 *GPIOs (General Purpose Input Output o Entradas y Salidas de Propósito General)*, 16 entradas analógicas con posibilidad de uso para *GPIOs*, un puerto USB y otro ICSP. Estos dos últimos puertos están destinados a la programación de la placa, volcando el programa a la memoria de esta. Además, se puede utilizar el puerto USB para comunicarse con la placa usando una comunicación serie, una vez traducida por un módulo FTDI incluido en la propia placa.

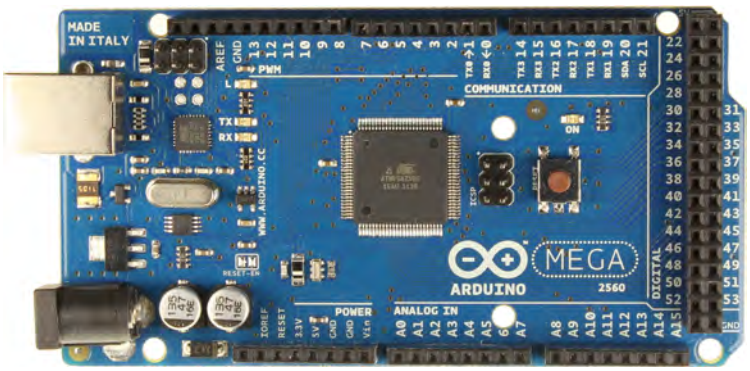


Figura 3.14: Arduino Mega 2560.

La frecuencia de reloj es de 16 MHz, generada por un oscilador de cuarzo. Al ser un microcontrolador, sólo se dispone de un único núcleo de procesamiento. Además se dispone de un único *hilo o thread* de procesamiento, que permite realizar una única tarea en ese instante de tiempo. A su vez implica que no se pueden programar tareas simultaneas de una forma convencional, debiendo de recurrir a otro métodos.

Este microcontrolador dispone además de 6 *Timer Array Unit o TAU*. Estos elementos son contadores que se actualizan cada ciclo de reloj. Esto, si se considera que la frecuencia de oscilación del cuarzo es siempre la misma, es el método perfecto para contabilizar tiempo. Estos contadores permiten poder crear interrupciones en el programa principal, pausando el *hilo* de procesamiento, ejecutar una función y volviendo a reanudar el programa por el punto por el que iba. Estos *timmer* también permiten generar señales digitales pulsantes, como señales *PWM*, sin que detengan o pausen el programa principal.

Además de la memoria reservada para guardar el programa, la placa dispone de 512 bytes de memoria EEPROM que permiten guardar unos pocos datos de forma permanente sin que se pierdan al dejar de alimentar la placa.

La placa trabaja a una tensión de $5V$, aunque es capaz de comunicarse con otros dispositivos digitales que funcionen a $3{,}3V$. Para alimentar la placa se pueden utilizar diferentes métodos, pudiendo aplicar la tensión de trabajo entre las patillas `5V` y `GND` o alimentándola a mayor voltaje a través de un *jack de alimentación* o usando los pines `Vin` y `GND`. La tensión de alimentación usando este método puede ir desde los 6 voltios hasta los 20 sin que se produzca deterioro. El consumo aproximado el microcontrolador es de 200 mA.

Al ser una placa que se puede replicar de forma legar al estar dentro de la filosofía de *Open Source*, en los que se liberan los diseños sin necesidad de una remuneración económica indicando únicamente de que autor proviene, el precio de estas placas es bajo al existir una gran oferta y demanda de ellas al estar orientadas a un público general no necesariamente especializado con la electrónica. Por estos motivos, se pueden encontrar replicas exactas de esta placa por un precio aproximado de 15 €, o si se desea una original por 50 €.

Por último, una ventaja de esta placa es que incluye su propio *software* para compilar y cargar los programas dentro de la placa, al incluir la *toolchain*. La programación de estas placas se realiza mediante un derivado del lenguaje *C++*. Este lenguaje admite casi todas las funciones que ofrece *C++* y permite la programación orientada a objetos. Todo esto facilita enormemente la tarea de programación a personas que no están familiarizadas con los microcontroladores. Además existe una amplia comunidad de usuarios que aportan sus conocimientos y avances a modo de librerías que se pueden implementar fácilmente en otros proyectos.

3.4.2. Raspberry Pi B

La siguiente placa que se muestra pertenece al grupo de los *SBC*. Esta placa ha nacido de un proyecto libre y está destinada en su origen al aprendizaje del campo de la informática, aunque posteriormente se ha ido abriendo paso en diferentes campos. Aunque permitan trabajar con electrónica de bajo nivel, como GPIOs, comunicaciones serial o IIC, estos dispositivos están más orientados a usarse como ordenadores de bajo coste orientados a la conexión a Internet y al mundo multimedia. Este tipo de placa puede soportar sin ningún problema un sistema operativo adaptado a su arquitectura, pudiendo instalar programas dentro de ella, además de gestionar periféricos de alto nivel como pantallas o dispositivos de audio. Para realizar estas tareas suelen llevar incluidos módulos auxiliares que permiten procesar tanto audio como vídeo sin ningún problema de rendimiento. Como se ha dicho, estos dispositivos están diseñados para utilizarse como pequeños ordenadores.

Raspberry Pi B está basada en el microprocesador *176JZF-S* a 700 MHz y presenta una arquitectura ARM v7 [9]. Aunque este microprocesador esté programado para utilizarse a esa frecuencia, la propia placa permite acelerar el reloj, sin crear daños en la placa ni perder garantías, pudiendo a llegar hasta un gigahercio.

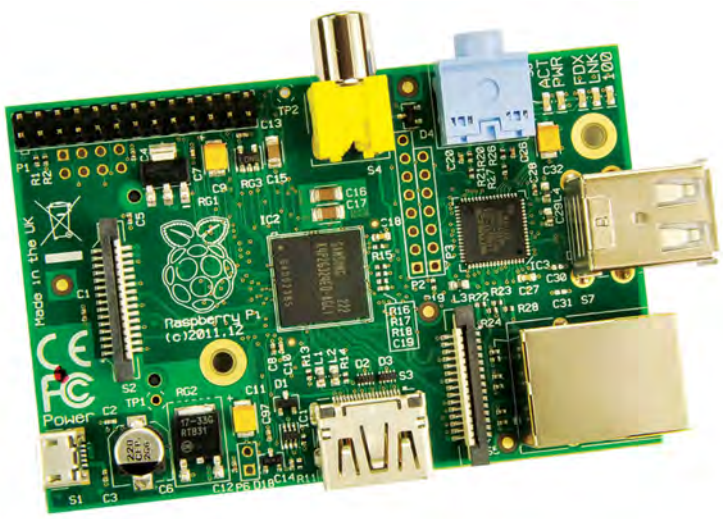


Figura 3.15: Raspberry Pi Modelo B.

Esta placa sólo dispone de 8 GPIOs, al mismo tiempo que permite usar comunicaciones SPI, IIC y UART en el resto de sus 24 pines. Esto hace que la placa quede bastante escasa de periféricos de bajo nivel. Además la placa incluye dos puertos USB, un conector HDMI, otro de salida compuesta y una salida estéreo de audio.

Esta placa necesita tener conectada una tarjeta SD para poder funcionar. Esto se debe a que no tiene memoria interna de estado sólido y, por lo tanto, necesita esta memoria externa para funcionar. En esta tarjeta se guardan tanto los programas como el sistema operativo que se desee usar. Además, la placa incluye 512 MegaBytes de memoria RAM.

La placa no incluye ningún reloj en tiempo real ni ningún otro elemento que contabilice tiempo, por lo que no se pueden generar señales digitales de forma precisa. No obstante, al poder usar un sistema operativo, se da la opción de poder utilizar diferentes *hilos* pese a sólo tener un núcleo.

La tensión de trabajo de la placa es de 5 voltios, pudiendo ser alimentada por los puertos USB o por un pin dedicado a ello. Además el consumo aproxima de la placa es de 700mA a una velocidad de reloj de 700 MHz.

El precio de esta placa es de aproximadamente 26 €más impuestos.

Por último, la placa permite diferentes sistemas operativos especialmente adaptados para ese microprocesador. Para poder usarlo es necesario instalarlo previamente en la tarjeta de memoria externa. Tras este paso, se pueden guardar programas dentro de la memoria de la placa, pudiendo compilarlo usando el microprocesador o usando un ordenador para ese propósito apoyándose en un compilador compatible con la arquitectura ARM y guardar el archivo en la carpeta correspondiente en la memoria externa que se vaya a emplear.

3.4.3. BeaggleBound Black

Esta placa también pertenece a el género *SBC* y está destinada a un uso algo más profesional, al poder trabajar indistintamente con componentes de alto y bajo nivel.

El microprocesador es esta placa es el *TI Sitara AM3359 ARM Cortex A8* que trabaja a una frecuencia de 1GHz [10]. Este componente tiene una arquitectura ARM versión 11 y también puede acelerar el reloj para aumentar la velocidad de procesamiento.

La placa dispone de 65 GPIOs, y algunas de ellas pueden ser destinadas a comunicaciones o a medir señales analógicas en un rango entre 0 y 1,8 voltios. Esta placa además presenta

un conector mini-HDMI y dos puertos USB, aunque uno de ellos está más orientado a la programación de la placa.

Figura 3.16: BeaggleBound Black.

La propia placa dispone de una memoria de 2 GB en la que se pueden guardar los programas y el sistema operativo, además de tener un conector para una tarjeta microSD. En cuanto a memoria volátil, la placa dispone de 512 MB de memoria RAM.

La placa dispone de una reloj que permite conocer el tiempo real si se utiliza una pila auxiliar. Además la placa incluye 4 TAUs que se pueden destinar para la generación de ondas en alguno de los pines digitales. Al poder instalar un sistema operativo, se permite tener diferentes procesos de forma simultanea gestionados por un sólo núcleo.

La placa trabaja a un voltaje de 5 voltios, pudiendo alimentarla a través de los puertos USB o usando el jack de alimentación. En un funcionamiento normal, la placa consume 500 mA. Es necesario indicar que no dispone de un regulador de tensión.

BeaggleBound Black presenta un precio aproximado de 49$, existiendo versiones superiores a un precio mayor.

Esta placa soporta diferentes sistemas operativos diseñados para este microprocesador y su arquitectura, pudiendo instalarlo directamente sobre la memoria interna de la placa. Además se pueden compilar y guardar otros programas dentro de la tarjeta de memoria o dentro de la placa.

3.4.4. Intel Galileo

Galileo es una placa microcontroladora basada en el chip *Intel Quark SoC X1000* [2] [13], basada en una arquitectura de 32-bit similar a la clase *Pentium*. Esta placa presenta la característica que dispone los pin en la misma posición que una placa *Arduino Uno* en su revisión R3, por lo que se pueden utilizar todos las placas de expansión diseñadas para esa plataforma. Este microprocesador tiene una frecuencia de procesamiento de 400 MHz. La placa disponen de los mismos pines que una placa *Arduino Uno R3* (14 GPIOs, 6 entradas analógicas y un puerto ICSP) y un puerto JTAG para funciones de depurado. Además se incluyen dos puertos USB y un conector Ethernet.

La placa dispone de 8 MB de memoria dedicada al programa base del la propia placa, haciéndola compatibles con el *software* de *Arduino*. Si se desean incluir programas más complejos, se pueden almacenar en la tarjeta micro SD, siempre que el programa esté preparado para la estructura del microprocesador. Existen características similares a la placa *Arduino Mega 2560* en cuento a memoria volátil y EEPROM, aunque en mayor cantidad. También presenta

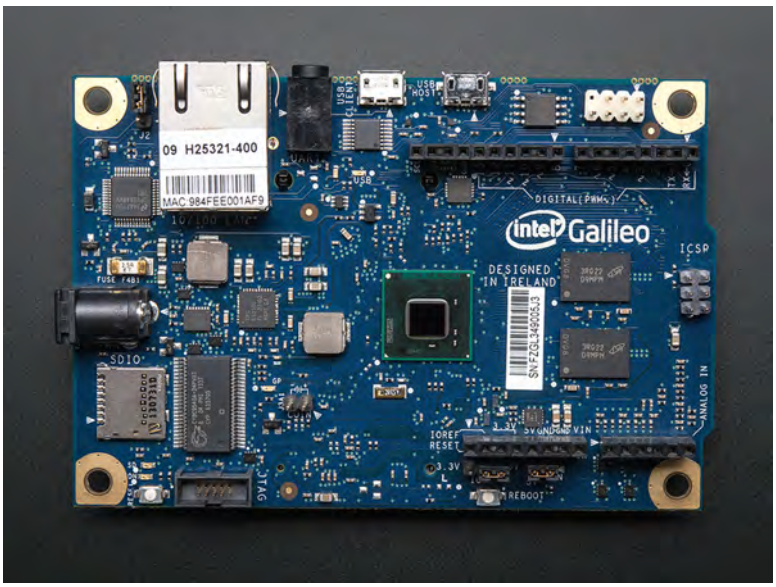


Figura 3.17: Intel Galileo.

4 TAUs dedicadas a la generación de señales PWM y comunicaciones ya citadas. Además la placa incluye un reloj a tiempo real.

La placa se ha de alimentar mediante un transformador AC-DC con una salida de 5 voltios y una corriente estable de hasta 3 amperios.

El precio aproximado de venta al público es de 60 €.

Por último, esta placa no soporta un sistema operativo como tal, si no que se basa en un *fireware* similar a las placas *Arduino*. Para programar esta placa se puede utilizar la entorno de *Arduino* pudiendo usar el mismo lenguaje y librerías que para otra placa.

3.4.5. RoBoard RB-110

La última placa de la que se hablará está totalmente diseñada y orientada a la robótica. Está preparada para soportar diferentes comunicaciones de bajo nivel al mismo tiempo que controla múltiples señales usando ondas PWM.

El microprocesador de esta placa es el *DM&P Vortex86DX* a 32 bit con compatibilidad para usarlo con x86. Este componente está preparado para usarse a 1000 MHz [18]. La placa dispone de numerosos puertos de bajo nivel que permiten conectar diferente elementos para comunicarse con ellos, sensarlos o cotrolarlos. Destacan los 16 puertos preparados para conectase directamente con un servo, al tener la misma configuración de pines que el conector del actuador. Además existen numerosos puertos para la comunicación por IIC, diferentes comunicaciones serial tanto a baja como alta velocidad, un puerto USB, un puerto JTAG, otro LAN y diferentes pins dedicados a convertidores analógicos-digitales.

La placa dispone de 256 MBde memoria RAM a bordo. Permite conectar una tarjeta micro SD donde poder leer y guarda datos relativos a la tarea a ejecutar.

La placa dispone de reloj a tiempo real, al mismo tiempo que incluye diferentes matrices de contadores de tiempo que permiten generar 16 señales PWM así como dar soporte a las comunicaciones de bajo nivel.

Se alimenta a 5 voltios y consume una corriente de 400 mA. Esta alimentación puede hacerse a través de dos pines destinados a este propósito o mediante un conector orientado a ello. La placa incluye un regulador de tensión, por lo que se puede alimentar a través de él desde 6 a 24 voltios.

Figura 3.18: RoBoard RB-110.

El precio de mercado de esta placa ronda los 270 $, pudiendo elegir entre otras versiones de esta gama a mayor precio.

3.5. Herramientas matemáticas para la robótica

En este apartado se intentará familiarizar el lector con las diferentes herramientas matemáticas en la que se apoya la base de la robótica. En muchas ocasiones es necesario orientar un sistema robótico en una posición del espacio y con una orientación específica para que pueda llegar a coger un objeto o posicionar la herramienta que porte en su extremo de un modo correcto. De esto aparece la necesidad de conocer la posición y orientación del punto objetivo con respecto al sistema de coordenadas del robot y poder variar los grados de libertad de este para que pueda alcanzar el objetivo. De aquí nacen las siguientes formas de expresar el entorno y poder controlar el robot. Todo lo explicado aquí está basado en el libro *Fundamentos de Robotica*[16], siendo este apartado una mera simplificación de un campo de las matemáticas muy extenso. Si fuese necesario, para una mayor comprensión de todo lo que se va a exponer, se puede recurrir al libro antes citado o cualquier otra publicación especializada para aumentar los conocimientos sobre este tema.

3.5.1. Sistemas de coordenadas

Depende si se quiere analizar un sistema en el plano o en el espacio, existen diferentes tipos de formas de definir un punto a un sistema de referencia. Estos sistemas de coordenadas tienden a homogeneizar el espacio en el que se implanten al poder definir con precisión distancias entre diferentes elementos usando una base espacial que se define igual para todos ellos.

Existen diferentes formas para indicar en que posición se encuentran los diferentes elementos en el plano o en el espacio. Para clasificarlos de un modo más sencillo se dividirán en dos grupos dependiendo de si se refieren a dos o tres dimensiones.

Sistemas de coordenadas planos

Coordenadas cartesianas El primero de los sistemas de referencias que se verán para analizar los diferentes posiciones de los elementos de un plano será el sistema de coordenadas cartesiano. Este sistema de coordenadas está referido a un punto y toma como directores de los ejes X' e Y' a dos vectores linealmente independientes entre ellos $\vec{u}$ y $\vec{v}$. En función de la relación de

la distancia al punto con el módulo, dirección y sentido a cada uno de estos dos vectores se definen las coordenadas de ese elemento.

Para simplificar los cálculos y facilitar la comprensión, los vectores $\vec{u}$ y $\vec{u}$ se eligen ortonormales y con módulo igual a la unidad, de tal modo que su producto vectorial sea perpendicular y saliente al plano, siendo estos $\vec{i}$ y $\vec{j}$. Esto transforma a los ejes X' e Y' en X e Y y origina una cuadrícula recta que permite una visualización más sencilla de los elementos.

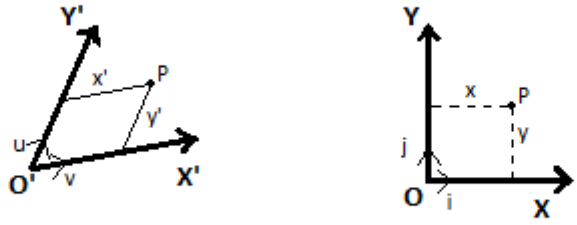


Figura 3.19: Sistemas de coordenadas cartesianos planos basados en $\vec{v}$ y $\vec{u}$, y en $\vec{i}$ y $\vec{j}$.

Coordenadas polares Este sistema de referencia está basado en definir los puntos en el plano usando una referencia de posición fija O, la distancia en línea recta que conecta a los dos puntos r y el ángulo θ que separa referencia con el segmento r con una línea que se tomó como referencia.

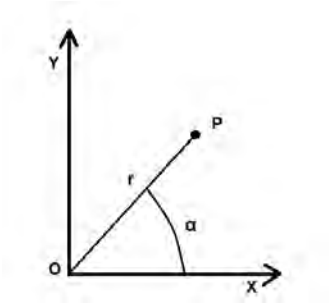


Figura 3.20: Sistema de coordenadas polares.

Sistemas de coordenadas espaciales

Coordenadas cartesianas Del mismo modo que ocurre en dos dimensiones, se puede definir el espacio usando tres vectores. Este tercer vector $\vec{k}$ suele ser siempre el producto vectorial de $\vec{i}$ y $\vec{j}$ y define al eje Z. Como se puede ver, los tres vectores tienen módulo unidad y son linealmente independientes, por lo que son apropiados para referir el espacio a ellos si ambos parten del mismo punto de referencia O.

Coordenadas cilíndricas Este sistema de referencia está basado en el sistema de coordenadas polares, añadiendo, para completar el espacio tridimensional, un tercer vector perpendicular al plano $\vec{k}$ que define al eje Z.

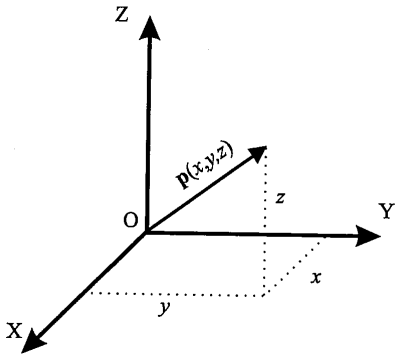


Figura 3.21: Sistema de coordenadas cartesiano tridimensional.

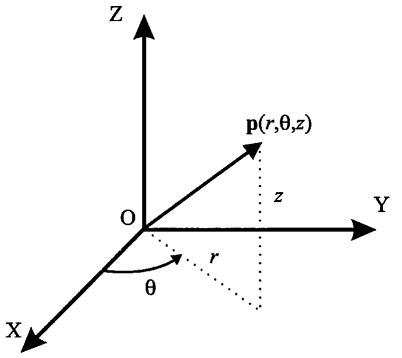


Figura 3.22: Sistemas de coordenadas cilíndricas.

Coordenadas esféricas Por último, este sistema de referencia se basa en medir desde un punto de origen O la distancia que separa ambos puntos r y los ángulos que origina. El primero de estos ángulo θ, hay que medirlo en la proyección en el plano vertical, estando definido por la proyección de la distancia r' y la línea de referencia. En cambio, el segundo ángulo ϕ está definido por la apertura entre r y el plano horizontal.

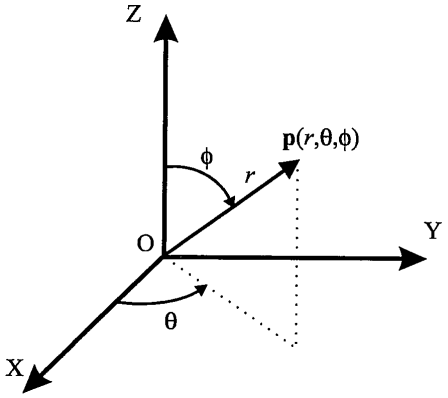


Figura 3.23: Sistema de coordenadas esférico.

3.5.2. Representación de la orientación

Con los sistemas de coordenadas se pueden representar puntos en el espacio o en el plano. En el caso de un cuerpo, también es necesario representar la orientación para que quede totalmente definido al contener diferentes puntos. Para ello es necesario representar el sistema de

coordenadas local de ese cuerpo junto con una relación angular que lo relacione con el sistema de coordenadas global.

Matrices de Rotación

Este sistema es uno de los más extendidos al basarse en matrices, por lo que se puede emplear fácilmente usando álgebra matricial. Este sistema permite relacionar un giro sobre uno de los ejes cartesianos con una matriz, que al multiplicarla por el vector referido a una orientación previa se obtiene un vector orientado según el ángulo girado.

Para saber el aspecto de esta matriz sólo es necesario analizar las proyecciones del nuevo sistema de orientación sobre el antiguo al rotar el primero sobre un eje del segundo. El eje de rotación es idéntico para ambos sistemas de referencia. Como se puede ver, al ser proyecciones provocadas por un giro, estas están expresadas por relaciones trigonométricas del tipo seno y coseno.

Dependiendo sobre que eje se gire, los elementos partícipes en la matriz se recolocan de un modo diferente, como se ve en la figura 3.24.

$$R(\alpha, x) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix}$$

$$R(\phi, y) = \begin{bmatrix} \cos\phi & 0 & \sin\phi \\ 0 & 1 & 0 \\ -\sin\phi & 0 & \cos\phi \end{bmatrix}$$

$$R(\theta, z) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

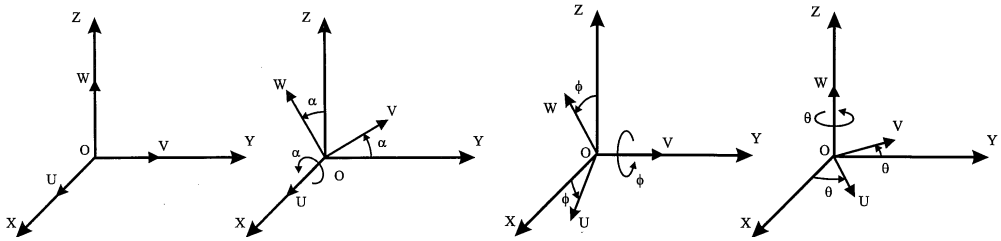


Figura 3.24: Giros en los diferentes ejes.

Como se ha visto hasta ahora, estas matrices sólo se pueden usar para representar la rotación en un solo eje. Si se desea rotar un cuerpo rígido sobre un ángulo que no está únicamente definido en un solo eje, es necesario multiplicar entre sí cada una de las matrices a las que la descomposición del ángulo a girar afecten a cada eje. Esto puede ocasionar operaciones del siguiente tipo.

$$T = R(\alpha, x) \cdot R(\phi, y) \cdot R(\theta, z)$$

$$T = R(\phi, y) \cdot R(\theta, z) \cdot R(\alpha, x)$$

Es necesario señalar que el orden en el que se decida descomponer el giro es muy importante, ya que el producto de las matrices no es conmutativo. Por lo que, si se desarrollaran las dos ecuaciones anteriores, la matriz de rotación final sería diferente para ambos casos.

Ángulos de Euler

Una representación sencilla de lo que se acaba de ver puede estar dado por estos tres ángulos, ϕ, θ y Ψ. Estos ángulos van referidos a unos ejes locales al cuerpo, que al girarlos en un determinado orden se consigue sobre el sistema de referencia local orientar al cuerpo en el espacio.

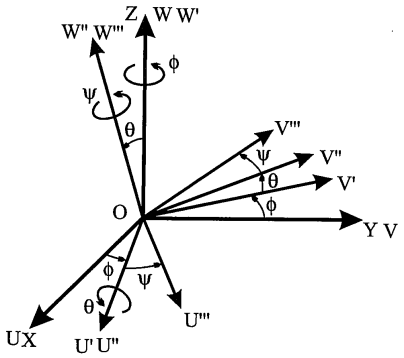


Figura 3.25: Giro representado por los ángulos de Euler.

Como se ha dicho antes, el orden en el que se giren estos ángulos es importante, al estar los siguientes giros referidos a los giros anteriores. Existen unos convenios ampliamente utilizados que permiten normalizar la orientación, al existir 24 posibilidades.

- **Ángulos de Euler ZXZ**
  En primer lugar, se rota el sistema de referencia local un ángulo ϕ sobre el eje Z, generando el sistema de referencia $X'Y'Z0$. Tras esto, se vuelve a hacer girar el sistema de referencia local sobre el eje X', que se ha desplazado en el giro anterior, un ángulo θ y que forma el sistema de referencia $X'Y''Z''0$. Por último, se vuelve a girar el sistema de referencia sobre el eje Z local del sistema de referencia Z'' un ángulo Ψ, obteniendo finalmente el sistema de referencia local la rotación deseada.

- **Ángulos de Euler ZYZ**

  Este método es idéntico al anterior, salvo que la segunda rotación se produce sobre el eje local Y' en lugar de hacerlo sobre X'. Es importante, si se va a usar los ángulos de Euler, decir cual de los dos criterios se ha de emplear, ya que la orientación final es diferente.

- ***Roll, Pitch and Yaw***

  Este sistema de referencia está muy extendido sobre todo en el mundo de la aeronáutica, al referir los ángulos a los ejes de un sistema fijo sin importar donde se encuentre el origen. Estos tres ángulos reciben el nombre en castellano de *alabeo, cabeceo y guiada*. Cada uno de estos ángulos está referido al giro sobre cada uno de los ejes X , Y y Z. Al igual que lo visto hasta ahora, es necesario indicar cual es el orden en el que se va a realizar los giros ya que no existe la propiedad conmutativa en el álgebra matricial.

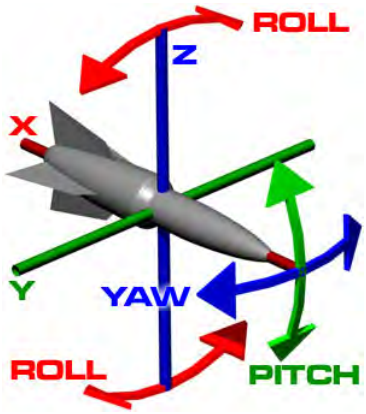


Figura 3.26: Representación de Roll, Pitch y Yaw.

Par de rotación

Para definir en giro en este sistema, sólo es necesario usar un vector $\vec{k}$ que indica el eje en el cual se ha de rotar el sistema un ángulo θ. Para que no se produzca traslación, el vector $\vec{k}$ ha de tener su origen en el mismo punto que el sistema de coordenadas de origen, para que el el nuevo sistema también lo está ahí.

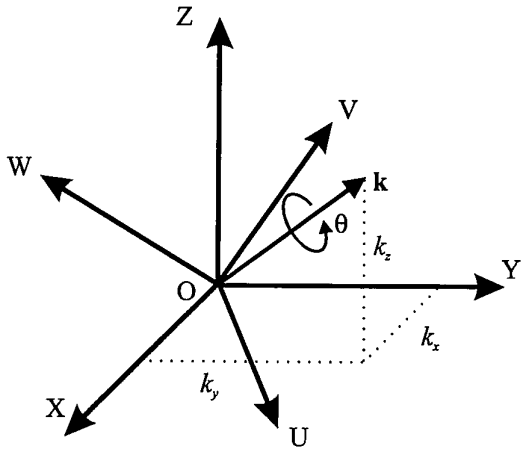


Figura 3.27: Representación de giro en torno a un vector.

El vector $\vec{k}$, al ser tridimensional, tiene tres componentes, por lo que este sistema esta definido por cuatro variables.

Cuaternios

Los cuaternios son una agrupación de cuatro elementos que definen la rotación de un sistema. Esta notación es similar a la anterior, y a que se define un ángulo que rota entorno a un vector generatriz.

Esta notación está muy extendida en robots comerciales, aunque en este proyecto no se empleará, por lo que no se profundizará en este sistema.

Relación entre las diferentes representaciones

Se pueden relacionar todos los métodos que indican la orientación de un cuerpo unos con otros. Pero por no ser materia de este proyecto, siendo esta sección una breve introducción al mundo de las herramientas matemáticas de la robótica, no de mostrarán en este este texto. Del

mismo modo, tampoco se relacionarán las diferentes formas de relacionar las traslaciones. En el caso de querer ahondar más en estos conocimientos, se recomienda leer otras publicaciones, como el ya citado libro *Fundamentos de la Robótica* [16].

3.5.3. Matriz de transformación homogénea

Hasta ahora se han definido los sistemas de referencia y las orientaciones por separado. Esta matriz responde a la necesidad de aunarlos en un solo elemento matemático. Si se considera la agrupación de las tres coordenadas espaciales con un cuarto elemento que represente la orientación, se puede multiplicar este vector por una matriz que rote y traslade al vector a una nueva posición y orientación. Sin entrar en el desarrollo matemático, estas matrices de $4x4$, al estar en un espacio tridimensional, están definidas por cuatro agrupaciones que representa una transformación del vector en la realidad.

$$T = \begin{bmatrix} R_{3x3} & P_{3x1} \\ f_{1x3} & w_{1x1} \end{bmatrix} = \begin{bmatrix} Rotación & Translación \\ Perspectiva & Escalado \end{bmatrix}$$

Al estar referidos a elementos del mundo real, la parte de la matriz que corresponde a la perspectiva o deformación de los elementos es cero, para no deformar la realidad. Del mismo modo, los elementos reales no sufren variaciones de volumen o forma, por lo que el valor de escalado suele ser siempre igual a uno. Si se toman estas consideraciones, las matrices homogéneas que se utilizan en robótica sólo suelen tener los términos de Rotación y Traslación.

$$T = \begin{bmatrix} Rotación & Translación \\ 0_{1x3} & 1 \end{bmatrix}$$

Siendo la *Rotación* la matriz correspondiente al giro y la *Traslación* un vector $\begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix}$ que esta expresado con las componentes divididas por coordenadas.

Esta matriz es la base de muchas herramientas matemáticas que se usan en robótica, al relacionar sistemas de coordenadas de diferentes elementos. Estas matrices se pueden multiplicar entre sí, pudiendo relacionar sistemas de coordenadas que tengan elementos móviles entre medias de ellos.

En ocasiones, estas matrices se suelen representar de la siguiente forma.

$$T = \begin{bmatrix} \mathbf{n} & \mathbf{o} & \mathbf{a} & \mathbf{p} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.1}$$

Siendo **n, o** y **a** vectores de tres coordenadas que representan la rotación en cada uno de los ejes y **p** que representa la traslación. Los tres vectores de la orientación son una terna ortonormal a derechas, **n** x **o** = **a**, a demás de tener cada uno el modulo unitario. Esto provoca que

$$\begin{bmatrix} \mathbf{n} & \mathbf{o} & \mathbf{a} \end{bmatrix}^{-1} = \begin{bmatrix} \mathbf{n} & \mathbf{o} & \mathbf{a} \end{bmatrix}^{T} \tag{3.2}$$

Esto permite obtener fácilmente la expresión de T^{-1}

$$T^{-1} = \begin{bmatrix} n_x & n_y & n_z & -n^T \cdot p \\ o_x & 0_y & o_z & -o^T \cdot p \\ a_x & a_y & a_z & -a^T \cdot p \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.3}$$

Esto permitirá poder sacar relaciones del tipo $\vec{r_{xyz}} = T \cdot \vec{r_{uvw}} \Rightarrow T^{-1} \cdot \vec{r_{xyz}} = \vec{r_{uvw}}$

3.6. Control cinemático

3.6.1. Problema Cinemático Directo

Dado un conjunto de articulaciones, prismáticas o rotativas, enlazadas por sus extremos entre sí de tal modo que un eslabón extremo se considere como la base del mecanismo, se puede relacionar el otro extremo con el sistema de referencia de la base. Al tener grados de libertad entre la base y el extremo, la posición y orientación final cambian en función de la posición y orientación los elementos intermedios.

El problema cinemático directo consiste en imponer un valor para cada grado de libertad y ver en que posición y orientación queda el extremo del mecanismo, referido a la base del mismo u otro sistema de referencia fijo.

Para solucionar el problema cinemático inverso es necesario hallar, en el caso de ser un problema en el espacio, ciertas ecuaciones que relacionen a los grados de libertad y elementos geométricos del robot con las coordenadas espaciales x, y y z y la orientación expresada como, por ejemplo, α, β y γ, como la relación de los ejes del sistema de referencia local y global.

Método geométrico

En ocasiones, en robot sencillo con unos pocos grados de libertad o teniendo sólo elementos en un plano, se pueden sacar estas relaciones basándose en la trigonometría.

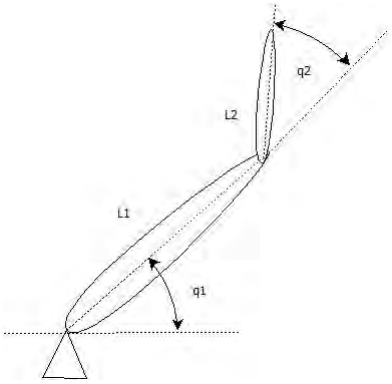


Figura 3.28: Mecanismo de ejemplo.

En este apartado se resolverá un pequeño mecanismo empleando este método. El robot a tratar está representado en la figura 3.28 y constan de dos elementos unidos entre sí por dos articulaciones rotativas. En primer lugar, es necesario conocer la posición y orientación que se obtendrá en el extremo del brazo provocando un giro en ambas articulaciones. Esto se puede representar como aparece a continuación.

$$x = l_1 \cdot \cos(q_1) + l_2 \cdot \cos(q_1 + q_2)$$

$$y = l_1 \cdot \sin(q_1) + l_2 \cdot \sin(q_1 + q_2)$$

$$\theta = q_1 + q_2$$

A través de estas ecuaciones, se puede conocer la posición y orientación del extremo, simplemente sustituyendo en ellas los valores necesarios.

Transformaciones Homogéneas

En el caso de robots con un número alto de grados de libertad, puede ser muy complejo el cálculo de estas relaciones usando ese método. Existe un método sistemático aplicable a cualquier brazo robótico con grados de libertad rotativos o prismáticos. Este método consiste en generar una matriz de transformación homogénea por cada grado de libertad. Esta matriz reproduce la rotación y traslación del sistema de referencia del extremo del eslabón más alejado con respecto al más cercano a la base. Cada una de estas matrices queda definida por ${}^{i-i}A_i$, siendo i un eslabón cualquiera perteneciente al mecanismo.

Al multiplicar en orden, partiendo de la base hacia el extremo, cada una de estas matrices se obtiene finalmente las coordenadas y orientación del extremo del brazo robótico.

$$T = {}^{0}A_1 \cdot {}^{1}A_2 \cdot {}^{2}A_3 ... \cdot {}^{n-i}A_n = {}^{0}A_n \tag{3.4}$$

Como se puede ver en la ecuación 3.4, este método no tiene un límite en el número de elementos que se pueden concatenar, obteniendo, tras multiplicar matriz a matriz, una solución real.

Algoritmo de Denavit-Hartenberg

En ocasiones no puede quedar muy claro como definir el sistema de coordenadas para cada elemento, pudiendo obtener de forma equívoca la matriz de transformación. Como solución a esto, Denavit y Hartemberg propusieron en 1955 [23] un algoritmo para obtener en cada elemento un sistema de referencia y la matriz de transformación homogénea de una forma sistemática.

Para obtener la matriz de transformación homogénea de cada elemento es necesario obtener otras cuatro matrices homogéneas que al multiplicarlas entre sí se obtenga la deseada. Al realizar siempre en el mismo orden unas ciertas traslaciones y rotaciones se puede obtener gráficamente una relación entre el sistema de coordenadas de un eslabón con el siguiente. Para obtener los parámetros que relacionan los sistemas de referencia basta con realizar en la matriz de transformación homogénea estas cuatro transformaciones.

1. Rotación sobre el eje z_{i-1} un ángulo θ_i para que los ejes x_{i-1} y x_i queden paralelos.
2. Traslación en dirección del eje z_{i-1} una distancia d_i para que ambos sistemas de referencia queden alineados.
3. Traslación en dirección del eje x_{i-1} una distacia a_i para que el origen de ambos sistemas coincidan.
4. Rotación sobre el eje x_{i-1} un ángulo α_i para que los sistemas de coordenadas S_{i-1} y S_i coincidan.

Si estas variaciones se trasladan cuatro matrices de transformación homogénea se puede sacar la relación genérica para todos los eslabones.

$${}^{i-1}A_i = \begin{bmatrix} \cos\theta_i & -\sin\theta_i & 0 & 0 \\ \sin\theta_i & \cos\theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & a_1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha_i & -\sin\alpha_i & 0 \\ 0 & \sin\alpha_i & \cos\alpha_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$${}^{i-1}A_i = \begin{bmatrix} \cos\theta_i & -\sin\theta_i\cos\alpha_i & \sin\alpha_i\sin\theta_i & a_i\cos\theta_i \\ \sin\theta_i & \cos\theta_i\cos\alpha_i & -\sin\alpha_i\cos\theta_i & a_i\sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.5}$$

Con esta matriz genérica basta con saber localizar los parámetros θ, d, a y α de cada eslabón. Para la obtención de cada parámetro de un modo correcto, es necesario plantear el *algoritmo D-H* para todo el brazo robótico, siguiendo en cada eslabón todos los pasos necesarios. Al tener una extensión considerable, no se detallará en profundidad este algoritmo, pudiendo encontrarlo en su libro de referencia [23].

Por último, sólo sería necesario incluir en cada matriz los parámetros correspondientes a su eslabón y multiplicarlas entre sí para obtener finalmente la cinemática directa.

3.6.2. Problema Cinemático Inverso

Del mismo modo que se pueden definir unos valores para cada grado de libertad de un brazo robótico y obtener la posición y orientación del extremo, siendo este el problema correspondiente a la cinemática directa, se puede definir una posición y orientación del extremo y obtener el valor que ha de tomar cada grado de libertad para conseguirlo. Esto último recibe el nombre de problema cinemático inverso.

Este es el problema que realmente interesa resolver en la robótica, al poder definir una posición y orientación en el espacio y que el extremo del brazo robótico pueda alcanzarlo. Basándose en la cinemática inversa de un brazo, se puede manejar a antojo a este elemento.

Al contrario que la cinemática directa, este problema puede presentar diferentes soluciones, generando diferentes vectores solución $\vec{q_n}$ que satisfagan los requerimientos. En ocasiones, es necesario limitar algunas de estas soluciones por no ser las más apropiadas, pudiendo limitar los recorridos de ciertos grados de libertad o imponiendo nuevas ecuaciones.

En ciertos casos, si el brazo robótico presenta unas características concretas, se puede analizar el mecanismo dividiéndolo en dos partes, una encargada del posicionamiento y otra de la orientación. Para que se pueda hacer esto, como se verá, es necesario no tener más de 6 ejes y que los tres últimos ejes se corten en un mismo punto. Esto provocará que los tres últimos ejes se encarguen de orientar el extremo, mientras que los restantes se encargarán del posicionamiento del punto de corte de esos tres ejes.

Método geométrico

Partiendo de las ecuaciones que definen la cinemática directa de un robot, se pueden obtener otras relaciones que permitan definir los grados de libertad en función de otros elementos y las posiciones y orientaciones requeridas.

Este tipo de método están indicado para robots con pocos grados de libertad o que están definidos en un plano, ya que su complejidad es menor. Cuando más grados de libertad tiene un mecanismo, más aumenta la dificultad para obtener la cinemática inversa al aparecer más parámetros que hay que interrelacionar. Por este motivo, este método sólo es recomendable usarlo cuando se tengan pocos grados de libertad y se pueda despejar fácilemente la solución partiendo de la cinemática directa.

Como ejemplo, se resolverá el mecanismo presentado en la figura 3.28 y se partirá de las ecuaciones del problema cinemático directo.

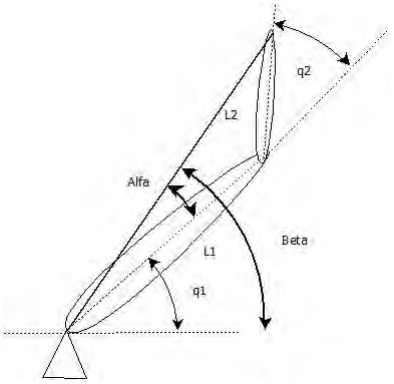


Figura 3.29: Resolución de mecanismo simple por método geométrico.

$$q_1 = \beta - \alpha$$

$$\beta = \arctan \frac{y}{\sqrt{x^2 + y^2}}$$

$$\alpha = \arctan \frac{l_2 \sin q_2}{l_2 \sin q_2 + l_1}$$

$$q_1 = \arctan \frac{y}{\sqrt{x^2 + y^2}} - arctan \frac{l_2 \sin q_2}{l_2 \sin q_2 + l_1}$$

$$q_2 = \theta - q_1$$

Por lo tanto

$$q_1 = \arctan \frac{y}{\sqrt{x^2 + y^2}} - arctan \frac{l_2 \sin(\theta - q_1)}{l_2 \sin(\theta - q_1) + l_1}$$

Se puede ver que la solución del problema depende de los datos de orientación y posición del extremo y de los parámetros geométricos del robot. En este casi, para resolver esta ecuación es necesario iterar, ya que la variable aparece a ambos lados de la ecuación.

Transformaciones homogéneas

En teoría se puede resolver la cinemática inversa partiendo de las transformadas homogéneas, al conocer la relación de los valores de los grados de libertad con los datos espaciales de salida. En la práctica esto no es tan sencillo, al obtener siempre más ecuaciones que los seis parámetros de la orientación y posición que se desean tener. Esto provoca que varias de estas ecuaciones estén interrelacionadas entre sí, pudiendo tener un sistema de ecuaciones no definido. En el caso favorable de que esto no ocurra, sería vital saber seleccionar que ecuaciones despejar y como relacionarlas con el resto. Una mala elección en ecuación de partida puede provocar que el cálculo sea complejo, pudiendo llegar a no resolverse.

En lugar de enfrentarse directamente a resolver el problema completo, se puede dividir en pequeños problemas que dependen de menos variables. Partiendo siempre de

$$T = {}^0A_1 \cdot {}^1A_2 \cdot {}^2A_3 \cdot \ldots \cdot {}^{n-i}A_n$$

Se puede obtener n sistemas de ecuaciones.

$${}^0A_1^{-1} \cdot T = {}^1A_2 \cdot {}^2A_3 \cdot \ldots \cdot {}^{n-i}A_n$$

$${}^1A_2^{-1} \cdot {}^0A_1^{-1} \cdot T = {}^2A_3 \cdot \ldots \cdot {}^{n-i}A_n$$

...

$$ {}^{0}A_{1}^{-1} \cdot {}^{1}A_{2}^{-1} \cdot \ldots \cdot {}^{(n-2)}A_{(n-1)}^{-1} \cdot T = {}^{(n-1)}A_{n} $$

Si uno se fija, se puede ver que a la izquierda del símbolo de igual se encuentra las variables $q_1 ... q_i$ quedando en la derecha $q_{i+1} ... q_n$. Esto permite obtener una mayor simplificación al obtener n sistemas con 12 ecuaciones cada uno[2]. Si se selecciona cuidadosamente, se puede elegir un elemento del cual se pueda despejar fácilmente la relación que se busca. Si se sigue un orden, por ejemplo empezando desde q_1 hasta q_n o viceversa, se pueden sacar las soluciones de cada grado de libertad dependiendo únicamente de los valores espaciales deseados.

Desacoplo Cinemático

En ocasiones, por motivos de diseño si tiene que los últimos tres de los seis grados de libertad cortan en un único punto. Esto puede permitir separar el problema de la cinemática inversa en dos partes. En primer lugar, los tres primeros grados de libertad se pueden encargar únicamente del posicionamiento del punto de intersección de los otros grados de libertad, por otro lado, los tres últimos grados de libertad permiten orientar fácilmente a ese punto como se quiera partiendo de los valores obtenidos para los tres primeros ejes.

Como se ha visto, esto sirve de solución para unos pocos casos, donde se produce la intersección de los ejes finales en un robot de no más de seis grados de libertad. En otras ocasiones, también se podrá plantear esta metodología, pero existirán más complicaciones al no cortarse los tres ejes.

3.6.3. Matriz Jacobiana

Además de tener las relaciones entre los elementos espaciales y las posiciones de los grados de libertad, interesa poder tener un control de velocidades para generar trayectorias. Por ejemplo, si se desea trazar una línea recta con el extremo del robot a velocidad constante, es necesario saber como varían las velocidades angulares en cada eje para que se cumpla esta condición. Para conseguir eso, es necesario incluir la expresión del tiempo en las variables. Para relacionar lo antes citado hay que recurrir a la *matriz Jacobiana*.

Para poder controlar la posición de un elemento, al trabajar con velocidades, sería necesario integrar. Si se integrase una trayectoria de velocidades se obtendría finalmente la posición que ocupa un elemento al saber la posición de inicio.

Matriz Jacobiana Directa

La *matriz Jacobiana directa* se obtiene a partir de derivar las ecuaciones de la cinemática directa respecto a cada variable una a una.

$$ J = \begin{bmatrix} \frac{\delta f_x}{\delta q_1} & \frac{\delta f_x}{\delta q_2} & \cdots & \frac{\delta f_x}{\delta q_n} \\ \frac{\delta f_y}{\delta q_1} & \frac{\delta f_y}{\delta q_2} & \cdots & \frac{\delta f_y}{\delta q_n} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\delta f_\gamma}{\delta q_1} & \frac{\delta f_\gamma}{\delta q_2} & \cdots & \frac{\delta f_\gamma}{\delta q_n} \end{bmatrix} $$

Con esta matriz se pueden relacionar las velocidades lineales en cada eje del espacio y las velocidades angulares de los ángulos de la orientación con las velocidades angulares de cada uno de los grados de libertad.

$$ \begin{bmatrix} \dot{x} & \dot{y} & \dot{z} & \dot{\alpha} & \dot{\beta} & \dot{\gamma} \end{bmatrix}^T = J \begin{bmatrix} \dot{q_1} & \dot{q_2} & \ldots & \dot{q_n} \end{bmatrix}^T $$

[2]En cada matriz están presentes los vectores **n, o, a** y **p** de tres elementos cada uno. Esto genera que existan 3x4 elementos, habiendo una ecuación para cada caso.

Es importante recordar que la matriz jacobiana depende de las posiciones de los grados de libertad en cada momento. Esto significa que varía dependiendo de la posición en la que se encuentre en cada instante. Por lo tanto, según se avance en una trayectoria, la matriz jacobiana cambia, teniéndola que recalcular en cada punto.

Esta matriz tiene, como es lógico, tantas columnas como grados de libertad. El número de filas depende de si se trabaja en el plano o en el espacio, o si se tienen en cuanta las orientaciones.

Matriz Jacobiana Inversa

Del mismo modo, también interesa crear trayectorias para poder controlar el extremo del robot. Para ello es necesario girar los grados de libertad a una velocidad diferente para cada instante. Para calcular esto, es necesario usar la *matriz jacobiana inversa*.

La *matriz jacobiana inversa* relaciona las velocidades lineales y las orientaciones deseadas con las velocidades angulares que hay que imprimir en cada motor para que se alcancen. Existe diferentes métodos para poder obtener la *matriz jacobiana inversa*.

El primer método consiste en invertir la *matriz jacobiana directa*. En ocasiones, aunque parezca sencillo, esta inversión pude resultar imposible. Suponiendo que haya tantos grados de libertad como parámetros, por ejemplo 6x6, obteniendo una matriz cuadrada, esta puede ser casi imposible de invertir al tener funciones trigonométricas complejas dentro de cada elemento.

Otra opción es resolver numéricamente la *matriz jacobiana inversa* imponiendo valores a cada q_n correspondientes a la posición actual, para obtener una matriz en la que sólo aparezcan números en lugar de ecuaciones. Tras esto sí se podría invertir la matriz fácilmente si fuese cuadrada. Podrían existir valores del vector q_n que hagan al determinante, o *Jacobiano*, igual a cero. Los valores que producen que el *jacobiano* se nulo se llaman *configuraciones singulares*. Estas se producen al quedar alineados dos o más ejes. Otro gran inconveniente que puede presentar esta *matriz jacobiana inversa* es que no sea cuadrada. En el caso de tener menos grados de libertad que el espacio de la tarea, el movimiento u orientación están restringidos. Como posible solución a esto, se puede eliminar un elemento del espacio de la tarea, al estar de por sí restringido, volviendo a obtener una matriz cuadrada. En el caso de que existan más grados de libertad que parámetros en el espacio, es decir, que el robot sea redundante, es necesario imponer que un grado de libertad tenga velocidad de rotación nula, dicho de otro modo, que el no pueda variar de posición. Para poder invertir esta, matriz no cuadrada es necesario recurrir a pseudoinversas como $\left(JJ^T\right)^{-1}$.

Al usar este método, se puede no alcanzar con precisión la posición y orientación deseadas. En la práctica resulta imposible calcular la *matriz jacobiana inversa* para cada punto del espacio. Para solucionar esto, se suele calcular una nueva *matriz jacobiana inversa* para un pequeña región del espacio. Los puntos de alrededor al que se calcula la matriz no tiene los mismos valores en los elementos pero si unos muy cercanos a ellos, por lo que se puede tomar como una aproximación, generando un pequeño error. Al integrar las aproximaciones de las velocidades, este error se propaga a la posición final.

Capítulo 4

Cinemática del robot MYOD

En este capítulo se pretende obtener las relaciones entre la posición del espacio en la que se encuentran las extremidades del robot con la rotación de cada una de sus articulaciones. Para ello se explorarán dos métodos para conseguir este objetivo. El primero de ellos está más orientado al control de un brazo robótico genérico y el segundo directamente orientado al robot. Se explican las desventajas que presentan los dos métodos y porqué se elige uno de ellos para dar solución al problema cinemático.

4.1. Cálculo de la cinemática por el método matricial

Para poder trazar trayectorias de caminata para el robot es necesario controlar de forma precisa las piernas. Es necesario tener algún tipo de herramienta que permita poder indicar a la pierna la orientación y posición que ha de alcanzar, teniendo que mover cada uno de sus elementos de forma correcta para adaptarse a esta imposición. Para ello, es necesario obtener la cinemática inversa, al considerar a cada una de las piernas como brazos robóticos separados. Al hacer esta consideración, se pueden utilizar diferentes herramientas matemáticas orientadas a la robótica para conseguir alcanzar el objetivo final.

Para poder obtener la cinemática inversa de un manipulador robótico es necesario seguir una serie de pasos por orden, pudiendo tener modificaciones para poder adaptarse al objeto de estudio dependiendo de sus particularidades.

4.1.1. Metodología

El método que se seguirá a continuación está basado en las matrices de transformación homogénea partiendo de los parámetros Denavit-Hartenberg. En primer lugar es necesario definir el elemento a analizar y aplicar el algoritmo D-H en él para poder conseguir relacionar los sistemas coordenados de la base del robot con el del extremo.

Tras sacar la relación directa del problema cinemático, es necesario obtener las ecuaciones que relacionan a la posición final con los giros del robot. Para ello, se necesitan despejarse una a una las matrices de transformación de cada eslabón, pudiendo obtener de alguno de sus elementos las ecuaciones que indiquen la cinemática inversa. Este paso se puede sustituir obteniendo las relaciones de la cinemática usando métodos geométricos, aunque puede resultar aún más complejo.

Después de obtener las relaciones de posición, es necesario hallar, apoyándose en estas relaciones, la matriz jacobiana propia del robot. Por último sólo es necesario invertir esta matriz para obtener la relación de control deseada.

Como la inversión de la matriz, al estar basada en relaciones trigonométricas, puede ser una tarea muy laboriosa, es mucho más sencillo sustituir en cada ecuación los valores de las

posiciones articulares en ese momento para obtener la matriz numérica. Esta matriz numérica es mucho más sencilla de invertir, facilitando en gran medida el cálculo.

La matriz jacobiana, al estar definida para velocidades en lugar de posiciones, no se puede usar directamente para asignar la trayectoria. Para poder definir una trayectoria, es necesario definir los puntos a alcanzar en función del tiempo. Tras hacer esto, basta con derivar la trayectoria respecto al tiempo para obtener las velocidades en cada punto. Partiendo del punto anterior, se puede obtener los valores numéricos para esa matriz obteniendo una aproximación suficientemente buena. Esta matriz hay que invertirla y multiplicarla por las velocidades obtenidas para ese intervalo. Esto permitirá obtener la velocidad de giro de los grados de libertad para esos instantes de tiempo y, que para poder aprovecharlo, debiendo integrar esa velocidad en el tiempo para obtener la variación en el giro. Partiendo de la posición inicial conocida para el punto anterior y el incremento angular que se ha calculado, se obtienen finalmente las posiciones de giro para ese instante, que además servirá como base para calcular la jacobina del siguiente punto de la trayectoria.

4.1.2. Obtención de las matrices de transformación por el método DH

Como se ha visto, el método de Denavit-Hartenberg [23] permite relacionar los sistemas de referencia de los diferentes elementos del robot aplicando un algoritmo sencillo. Este algoritmo permite obtener los parámetros geométricos que permiten relacionar un sistema de referencia con el siguiente. Si se considera la pierna del robot por separado y tratándola como se fuese una manipulador robótico cualquiera, se puede aplicar este algoritmo sin ninguna dificultad.

Por no tener que realizar el mismo trabajo dos veces, se puede analizar únicamente la pierna derecha y poder extrapolar los resultados a la izquierda por ser simétrica. En este estudio, es necesario volver a señalar que existen dos versiones diferentes de la pierna para el robot, por lo que es necesario analizar a cada una de ellas por separado.

4.1.3. Pierna derecha de la versión ancha

Para comenzar con el algoritmo, es necesario definir el sistema de referencia fijo y numerar cada uno de estos elementos. Esto ya se ha hecho previamente en la sección 2.2.1 siguiendo el mismo criterio que el que hay que utilizar para aplicar el algoritmo DH.

Tras esto, es necesario de indicar cual es el eje de actuación para cada articulación. En este caso, al ser todos los elementos rotativos, ese elemento coincide con el eje de rotación entre cada eslabón. Este eje, además hará de las veces de eje Z, pudiendo elegir arbitrariamente hacia donde apunta. Generalmente interesa que si existen diferentes ejes perpendiculares a un mismo plano, todos los ejes apunten en la misma dirección.

Después de esto, siguiendo el criterio impuesto por el algoritmo [23], se ha de situar el origen de referencia para cada eslabón sobre cada eje Z según se indique. Tras esto, se sitúa el eje X partiendo del origen ya colocado, siguiendo otro paso del mismo algoritmo. Al quedar definidos los ejes X_i, Z_i y el origen S_i de cada eslabón, de puede obtener fácilmente los ejes Y_i al considerar el sistema como dextrógiro.

Una vez que se han definido todos los elementos partícipes, se pueden empezar a obtener los cuatro parámetros que definen la relación entre los sistemas de coordenadas solidarios a cada eslabón. Como se mostró en la sección 3.6.1, los parámetros están definidos por rotaciones y desplazamientos a la largo de los ejes de su sistema de referencia que lo relacionan con el siguiente, siendo un orden determinado.

1. Rotación sobre el eje z_{i-1} un ángulo θ_i para que los ejes x_{i-1} y x_i queden paralelos.
2. Traslación en dirección del eje z_{i-1} una distancia d_i para que ambos sistemas de referencia queden alineados.

3. Traslación en dirección del eje x_{i-1} una distacia a_i para que el origen de ambos sistemas coincidan.

4. Rotación sobre el eje x_{i-1} un ángulo α_i para que los sistemas de coordenadas S_{i-1} y S_i coincidan.

Estos parámetros permitirán crear la matriz de transformación que relaciona ambos sistemas.

$$ {}^{i-1}A_i = \begin{bmatrix} \cos\theta_i & -\sin\theta_i\cos\alpha_i & \sin\alpha_i\sin\theta_i & a_i\cos\theta_i \\ \sin\theta_i & \cos\theta_i\cos\alpha_i & -\sin\alpha_i\cos\theta_i & a_i\sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Como se puede ver, cada una de estas matrices tiene en su interior las variables correspondientes a ese eslabón y los elementos geométricos que definen al mismo. Al multiplicar todas ellas en el orden correcto, se obtiene finalmente la matriz de transformación homogénea que relaciona el sistema de coordenadas de la base con el del extremo, teniendo en cuenta la posición de todas la variables presentes en el brazo.

Esto se puede aplicar a la pierna derecha de la versión que nos ocupamos, pudiendo obtener los siguientes parámetros. Para simplificar la notación, al ser todas las articulaciones rotativas, es necesario incluir el valor de giro de cada articulación en el valor θ_i sumándose al número mostrado en las tablas 4.1 y 4.2.

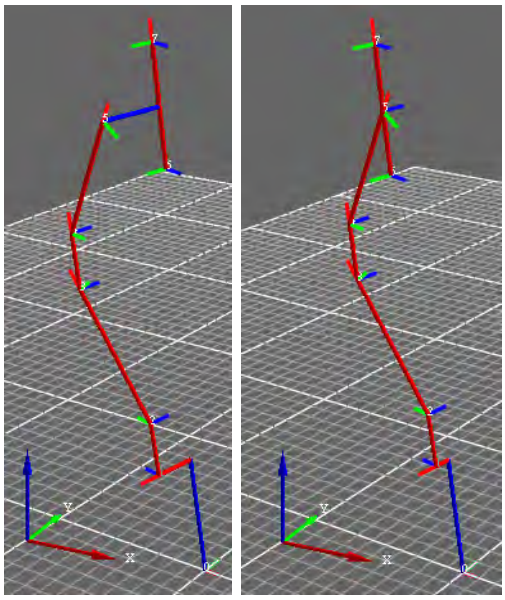
	θ_i	d_i	a_i	α_i
Eslabón 1	0	64.4	18	90
Eslabón 2	90	0	-27.5	90
Eslabón 3	0	0	69	0
Eslabón 4	0	0	27.5	0
Eslabón 5	0	0	69	0
Eslabón 6	0	27.5	-27.5	-90
Eslabón 7	0	0	55	0

Tabla 4.1: Parámetros DH para la versión de cadera ancha.

Estos parámetros definen cada una de las matrices de intermedias ${}^{i-1}A_i$ que al multiplicarlas entre sí de la forma correcta permiten obtener la cinemática directa de la pierna.

4.1.4. Pierna derecha de la versión estrecha

Del mismo modo, se puede aplicar el mismo algoritmo para obtener los parámetros de esta pierna. Se puede ver en la figura 4.1 la representación gráfica de los parámetros calculados.



(a) Versión Ancha (b) Versión Estrecha

Figura 4.1: Representación gráfica de los parámetros DH.

	θ_i	d_i	a_i	α_i
Eslabón 1	0	64.4	7.5	90
Eslabón 2	90	0	-27.5	90
Eslabón 3	0	0	69	0
Eslabón 4	0	0	27.5	0
Eslabón 5	0	0	69	0
Eslabón 6	0	0	-27.5	-90
Eslabón 7	0	0	55	0

Tabla 4.2: Parámetros DH para la versión de cadera estrecha.

4.1.5. Relacionar la transformada homogénea con las ecuaciones de la cinemática

Una vez que se tienen los parámetros correspondientes y la matriz de transformación, el siguiente paso consiste en la obtención de las seis ecuaciones que permitan obtener la matriz jacobiana.

Para ello, basándose en $T = {}^0A_1 \cdot {}^1A_2 \cdot {}^2A_3 \cdot \ldots \cdot {}^{n-i}A_n$ y lo visto en 3.6.2, se pueden ir despejando ordenadamente las diferentes matrices de cada eslabón para poder obtener las soluciones. Una vez que se han invertido y multiplicado las matrices, es pueden obtener, si se elige cuidadosamente el elemento de la matriz, las relaciones que se producen entre el espacio tarea y los grados de libertad.

$$ {}^0A_1^{-1} \cdot T = {}^1A_2 \cdot {}^2A_3 \cdot \ldots \cdot {}^6A_7 $$

$$ {}^1A_2^{-1} \cdot {}^0A_1^{-1} \cdot T = {}^2A_3 \cdot \ldots \cdot {}^6A_7 $$

$$ \ldots $$

$$ {}^5A_6^{-1} \cdot \ldots {}^1A_2^{-1} \cdot {}^0A_1^{-1} \cdot T = {}^6A_7 $$

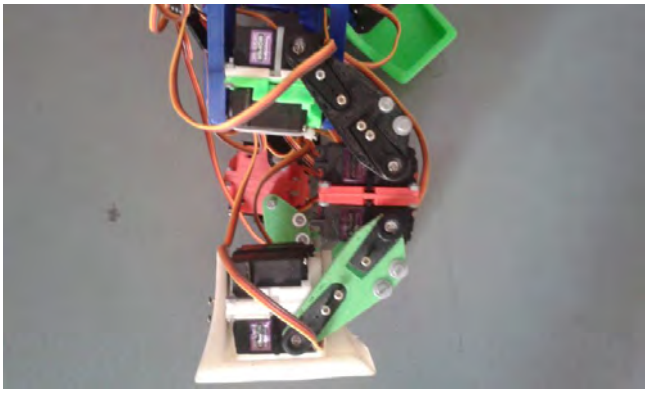
Como se puede ver, al tener siete grados de libertad se obtendrían siete sistemas de ecuaciones matriciales para poder obtener los seis parámetros del espacio tarea. Además, existiría un

grado de libertad más que el número de parámetros del espacio solución. Esto significa que la matriz jacobiana no será cuadrada al tener que existir siete derivadas parciales, teniendo unas dimensiones de [6x7]. Este hecho manifiesta que existen redundancias en el sistema, pudiendo existir diferentes soluciones para una misma situación.

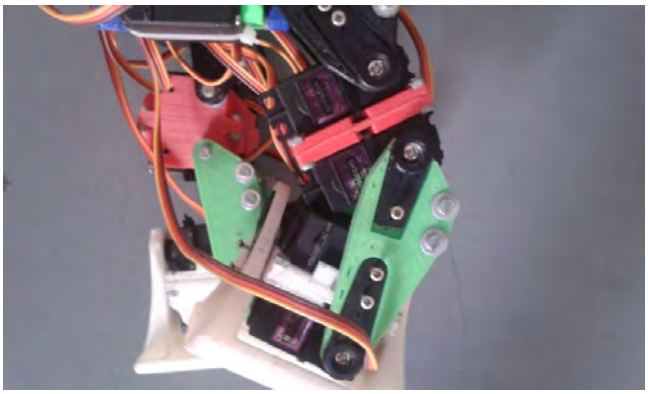
Esta condición de salida no es favorable para los intereses del proyecto, al tener que imponer alguna restricción para eliminar esta redundancia.

Para poder utilizar este método, y finalmente obtener una matriz [6x6], es necesario eliminar un grado de libertad del problema. El robot, por motivos de diseño en el cálculo de torque de la rodilla, incluyen dos servomotores q_4 y q_5 para dividir el esfuerzo total entre ambos. Esta pareja de motores son los causantes, al tener que meter dos grados de libertad donde sólo debería ir uno, de que exista la redundancia por definir la pierna con siete grados de libertad.

Si se quiere solucionar esto, puede ser necesario tener que condenar el movimiento de uno de estos grados de libertad. La condición que se le imponga a la pierna no puede tomarse a la ligera, ya que el robot presenta espacios pequeños entre los diferentes elementos móviles pudiendo originar colisiones, como se ve en 4.2. La pierna del robot está diseñada para que el eslabón de la rodilla siempre sea paralelo a la línea que une los ejes q_3 y q_6, evitando que choquen las carcasas de los servomotores entre sí.



(a) Sin colisión.



(b) Con Colisión.

Figura 4.2: Posibles colisiones entre motores.

Este motivo de diseño hace pensar que no se puede condenar el movimiento de un solo motor de la rodilla, ya que un giro moderado del otro eje puede generar el impacto de varios elementos. Por lo tanto es necesario descartar esta condición.

Por otro lado, se puede imponer una condición que indique que el eslabón 4 mantenga siempre la misma orientación, haciendo que siempre sea normal al suelo. Para ello, la suma angular de los ejes que orientan este elemento sea cero, siendo este valor el correspondiente

a la vertical. Si se analiza la pierna, la articulación q_1 no varía la orientación del eslabón 4 en relación la abertura entre ese eslabón y el plano horizontal. El eje q_2 varía la orientación del cuarto eslabón, pero también lo hacen en el tercero. La posible colisión se produciría al entrar en contacto los motores que mueven las articulaciones q_2 y q_4. Si se mueve el segundo eje, cambiará la orientación de el resto de ejes que estén a continuación. Por lo tanto, al no existir otro elemento que varíe la orientación hasta el eje q_7, los ejes intermedios seguirán siendo paralelos entre sí.

Por último, el tercer eje sí que altera la orientación relativa entre los motores, provocando esta articulación. Por supuesto, el cuarto eje también varía la orientación del cuarto eslabón. Por lo tanto, la suma de estos dos ejes ha de ser cero, para evitar las posibles colisiones. Este es un problema de diseño ya que este está orientado a mover los motores de esta forma, no permitiendo todas las combinaciones posibles de movimiento. Se puede considerar que al mover la pierna como un brazo robótico se ha de permitir elección en cuanto a la orientación del extremo. Si se consideran movimientos en los que la orientación referida a este plano no es excesivo, no colisionarían los motores del tobillo con los de la rodilla. Es necesario tener precauciones con esto ya que, aunque se permite el movimiento matemáticamente, la estructura puede hace chocar estos elementos en movimientos muy amplios.

$$q_3 + q_4 = 0$$

$$q_3 = -q_4$$

Esta condición sí que puede ser válida para el problema, ya que se reducen las variables a q_1, q_2, q_3, q_5, q_6 y q_7, por poder representar a q_4 en función de q_3. Esto además provoca que la matriz jacobina se transforme en una matriz cuadrada [6x6].

4.1.6. Desventajas de este método para la implementación en este proyecto

Aunque este método este muy extendido en la robótica, para este proyecto puede no ser viable. Esto es debido a la velocidad de procesamiento de la controladora. Esto hace que el tiempo total para desarrollar la tarea puede ser más alto que el deseado, no pudiendo generar trayectorias a tiempo real si la controladora es lenta.

Para poder utilizar este método es necesario recalcular e invertir la matriz jacobiana en cada punto de la trayectoria. Esta tarea, que puede ser sencilla, puede consumir un tiempo significativo por la controladora al ser una matriz [6x6], en el caso de que la imposición esté presente. Si además hay que considerar que hay que hacer el mismo procedimiento para la otra pierna, el tiempo total de cálculo se dobla. Si se considera que el robot ha de hacer otras tareas además de mover solo las piernas, como por ejemplo tomar mediciones de sus sensores, el tiempo total para procesar un punto de la trayectoria puede aumentar aun más al atender a una posible interrupción.

Además, al tener alineados los ejes q_3, q_4, q_5 y q_6 cuando la pierna está totalmente estirada, puede provocar singularidades. Además, el espacio de soluciones posibles al levantar el tobillo verticalmente es doble. Esto significa que pueden existir dos soluciones al considerar la posibilidad de *codo arriba* y *codo abajo* en la pierna, teniendo que descartar una de ellas para que el movimiento parezca humano, desplazando la rodilla hacia adelante.

Por estos motivos, al usar una controladora poco potente y querer diseñar un robot autónomo con el control a bordo, es necesario descartar este método para conseguir generar las trayectorias de caminata a tiempo real.

Por otro lado, esto se podría implementar si se calculase previamente las trayectorias de forma externa por un ordenador u otra máquina de computo similar. Esto, al considerar la trayectoria de caminata como algo repetitivo, se puede calcular mediante la jacobina una trayectoria completa correspondiente a un único periodo. Estas soluciones se podrían guardar en

un fichero de código. Cada vez que se demande usar la trayectoria, la controladora sólo ha de leer este fichero para mover la pierna según lo descrito antes.

No obstante, al considerar a las dos piernas como dos brazos robóticos unidos por la cadera esto sería necesario hacerlo por duplicado, complicando el código final. No sólo se complicaría el código si no la estabilidad, al tener que definir dos trayectorias sincronizadas que permitan la estabilidad del robot en todo momento. Esta sincronización puede ser otro punto en contra para este proyecto, al tener que hacer un estudio dinámico para el robot y subdividirlo en dos trayectorias.

Con relativa frecuencia, al despejar las ecuaciones de la cinemática directa, estas pueden estar relacionadas entre sí. Estas relaciones pueden hacer que el cálculo para resolver la ecuación necesite iteración. Esto significa que ha de resolver una misma ecuación un número alto de veces incluyendo en ella el valor antes calculado hasta la convergencia del dato. En ocasiones, si el extremo del brazo se encuentra cerca de un punto singular, el número de iteraciones necesarias puede aumentar, tardando aún más tiempo para calcularlo.

Como se ha dicho, los motores no tienen ningún tipo de sensor de par ni tampoco se dispone de ningún elemento en el robot que mida aceleraciones o giros del centro de gravedad por lo que no se pueden hacer correcciones a tiempo real del modelado matemático calculado externamente. Esto implica que el modelado ha de ser muy preciso debiendo incluir la mayor cantidad de variables posibles.

Aunque esto pudiera parecer viable, en la práctica exige una gran complejidad, teniendo que recurrir a programas especializados para el modelado del robot, el análisis de la cinemática inversa y las trayectorias de ambas piernas. Por esto no se va a usar finalmente este método aunque sea viable, dejándolo sólo indicado.

4.2. Estudio cinemático por superposición de movimientos

El método que se explicará a continuación está basado en la superposición de movimientos. Este método parte de modelos trigonométricos al dividir las articulaciones de las piernas del robot en subconjuntos que se analizan por separado. Al sumar el giro producido cada movimiento en las diferentes articulaciones se halla el valor total del giro que ha de realizar dicha articulación.

4.2.1. Introducción

En elementos robóticos sencillos, como brazos con dos o tres grados de libertad, es sencillo sacar la cinemática directa analizando matemáticamente las diferentes articulaciones rotativas o prismáticas. En ocasiones también es sencillo, despejando las ecuaciones de la cinemática directa, poder sacar las funciones que definen la cinemática inversa. Usando esta cinemática inversa, se puede posicionar el extremo del robot como se prefiera. Desgraciadamente esto es sólo posible en elementos muy sencillos, al complicarse exponencialmente el nivel de cálculo al aumentar los grados de libertad, sin saber si puede existir o no una solución. Por ello este método no se suele emplear en brazos robóticos con más de cuatro grados de libertad [16].

En el caso de querer usar este método analítico en una de las piernas del robot MYOD aparecerían incontables complicaciones, al disponer de siete grados de libertad en cada una de ellas. Al intentar sacar la cinemática directa del robot se tendría un sistema de ecuaciones complejo, con un número elevado de parámetros al tener que incluir en la fórmula los correspondientes a los elementos geométricos y las variables de cada articulación. Con esta complicación inicial, sería casi imposible sacar la cinemática inversa usando ese sistema de ecuaciones.

Además, al tener cuatro ejes consecutivos totalmente paralelos en cada pierna es casi natural que existiesen redundancias. Para evitar esto, habría que recurrir a ecuaciones de condiciones de contorno impuestas siguiendo algún criterio que restrinjan el espacio solución.

Para solucionar esto y poder usar una técnica similar, se ha decidido dividir la pierna en mecanismos más sencillos para analizarlos por separado. Para poder ser aprovechables los resultados de estos estudios, conviene que los movimientos estén separados en diferentes planos o, si se prefiere, en posibles movimientos sencillos. Posteriormente estas cinemáticas se podrán parametrizar para conseguir los movimientos deseados.

Una vez que se analicen los subconjuntos por separado es necesario volver a enlazarlos entre sí para reconstruir la movilidad total de la pierna. Para volver a obtener de nuevo el conjunto es suficiente con superponer los diferentes estudios hechos por separado. La forma para hacer esto es sumar a cada eje todas la variaciones de posición que le afecten diferentes estudios. Por ejemplo, si en dos estudios diferentes se incluye el mismo eje de la cadera, se han de sumar las variaciones de posición que provocan ambos movimientos a la posición inicial del eje.

$$\Delta q_{i_{movimiento}1} + \Delta q_{i_{movimiento}2} + ... + \Delta q_{i_{movimiento}n} = \Delta q_{i_{total}}$$

Como se verá, el mismo subconjunto se puede analizar de diferentes maneras para obtener distintas formas para mover dicho subconjunto, pudiendo incluir nuevos grados de libertad o restricciones que modifiquen el análisis siempre respetando los movimientos posibles en el robot. Esto puede permitir una gran libertad en las formas en las que se puede hacer andar el robot.

La forma de analizar estos subconjuntos se basa en usar relaciones trigonométricas que relacionen todos los ejes que formen ese movimiento con un parámetro que se pueda variar. El cambio de magnitud de este valor permitirá el posicionamiento de este conjunto en cierto lugar del espacio. En todos los caso, dichas ecuaciones dependen de elementos geométricos y de las diferentes posiciones que toman todos los grados de libertad. Si el parámetro maestro cambia a lo largo del tiempo, se pueden conseguir diferentes posiciones creando una trayectoria para ese conjunto.

Todas las ecuaciones de cada pierna están basadas en relaciones trigonométricas desarrolladas a partir de cada conjunto. Por este motivo, todas ellas se pueden calcular analíticamente. Esto es una ventaja ya que siempre se encuentra una solución y de una forma rápida, al obtener el resultado sustituyendo valores y no tener que iterar un número elevado de veces hasta la convergencia del dato. En el caso de tener una controladora con una velocidad de computo no muy elevada, este método se podría implementar dentro de ella estando a bordo del robot pudiendo generar y corregir nuevas trayectoria en función de la posición del robot o de los esfuerzos estáticos y dinámicos presentes en el cuerpo móvil.

4.2.2. Presentación de los subconjuntos

Para comenzar el estudio cinemático, es necesario definir cuales son los conjuntos en los que se va a dividir el cuerpo del robot. Cada uno de ellos se deberá analizar por separado, sin pensar en las posibles interferencias que produzca unos en otros.

En este robot con siete grados de libertad en cada pierna, con la particularidad de tener dos ejes de rotación en la rodilla, se dividirá el análisis en posibles movimientos diferenciados. Estos movimientos se pueden entender como una descomposición del movimiento de una pierna antropomórfica en movimientos más sencillos y que se pueden distinguir fácilmente.

Si se quiere levantar un pie del suelo, es necesario distribuir todo el peso del cuerpo sobre la pierna opuesta. Al hacer esto, la pierna liberada sólo soporta su propio peso, teniendo que hacer menos esfuerzos para poder separar el pie del suelo. Para conseguir repartir el peso entre una y otra pierna cuando los humanos andan, necesitan balancear la masa cuerpo. Este

balanceo produce unas aceleraciones en el cuerpo que permiten desplazar el centro de gravedad lateralmente para que quede alineado sobre una de las piernas, soportando la totalidad del peso sobre ella. Para imitar este desplazamiento de fuerzas se definirá este balanceo como uno de los subconjuntos. Como particularidad, este subconjunto estará formado por alguno de los ejes de las dos piernas y el eslabón de la cadera para unir ambas extremidades. Para denominarlo de algún modo para hacer referencia a él, este subconjunto será llamado desde ahora *movimiento de balanceo* y se analizará desde el plano de alzado. Además se puede ver que motores participan en este movimiento en la figura 4.3.

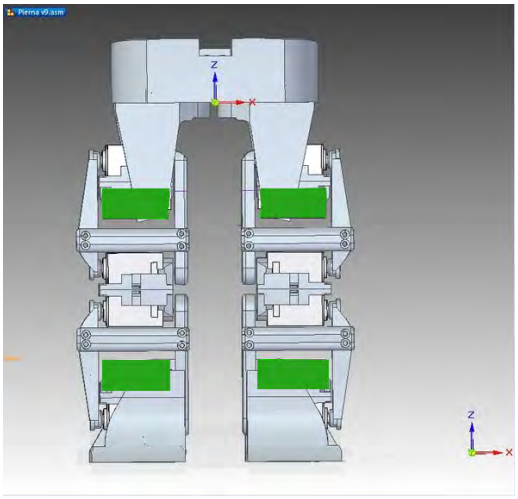


Figura 4.3: Motores que participan en el movimiento de Balanceo.

Una vez que se reparte el peso de un cuerpo sobre una pierna, la otra queda preparada para poder levantar el pie del suelo. Para que se produzca este levantamiento, se ha de disminuir la distancia total en línea recta desde el pié a la cadera. La manera en la que se hace esto es encogiendo la rodilla, a demás de mover la cadera y el tobillo de forma sincronizada para que el pie se mueva verticalmente sin que pierda su orientación. Es importante volver a señalar que también es necesario mover la pierna desde la cadera, ya que si no el extremo no se desplazaría verticalmente. Del mismo modo, el giro del tobillo provoca que la planta del pie quede paralela al plano horizontal. Las articulaciones recién citadas serán las que estén involucradas en este subconjunto, que se pueden ver en la figura 4.4,y será llamado *movimiento de rodilla*. Este estudio se realizará solo sobre el plano lateral, analizando por separado cada pierna.

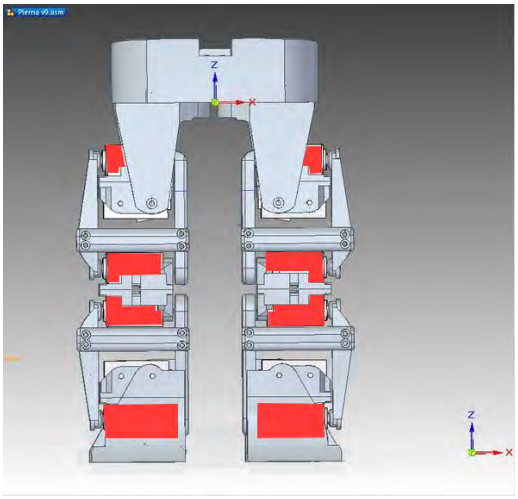


Figura 4.4: Motores que participan en el movimiento de Rodilla.

Para provocar el movimiento de avance del cuerpo móvil es necesario, junto a los *movimientos de rodilla y balanceo*, generar una fuerza resultante para que su reacción genere dicho avance. Para ello, es suficiente con desplazar hacia adelante y atrás el extremo de la pierna. Para este movimiento se han de girar los grados de libertad de la parte superior de la pierna y del tobillo que provocan que la pierna se desplace en el plano de perfil, como se ve en 4.5. Este es el tercer subconjunto, llamado *movimiento de avance*, que junto a los dos anteriores definen el movimiento básico para poder mover a un bípedo.

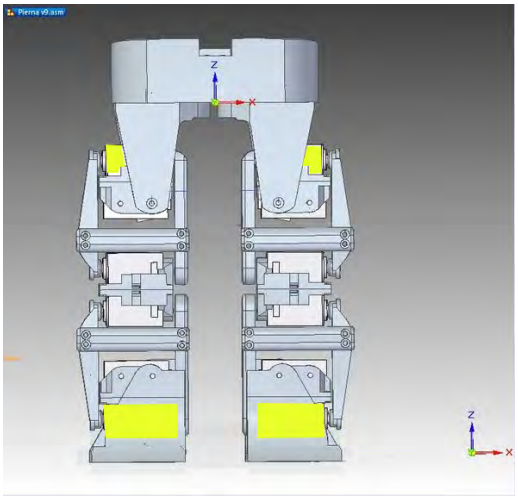


Figura 4.5: Motores que participan en el movimiento de Avance.

Los humanos al andar tiene más movimientos que hacen más complejo el movimiento total. Es muy difícil analizarlos todos, por lo que sólo se detallarán dos más. Estos dos movimiento no son totalmente indispensables para producir la caminata del robot, pero ayudan a imitar en cierta medida la forma en la que nos desplazamos los humanos.

El primero de estos dos movimientos es el que produce la rotación relativa del torso con la cadera. Este desplazamiento relativo permite emular el movimiento de la cadera, avanzado cada extremo de ella cuando lo haga la pierna correspondiente. Para imitar este movimiento no sólo se moverá la cintura, si no que también lo harán los motores que controlan la orientación de las piernas. En menor medida también lo hará la articulación del cuello, oponiéndose al giro de la cintura para orientar la cabeza siempre hacia adelante. Todos los motores participantes en este movimiento están señalados en la figura 4.6. Para cerrar esta definición, a este subconjunto se le llamará *movimiento de rotación de cintura.*

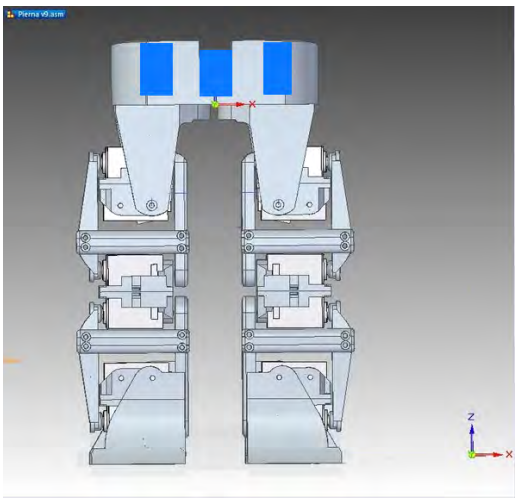


Figura 4.6: Motores que participan en el movimiento de Cintura(Sin motor del cuello).

Por último, y para concluir con el desglose de los subconjuntos, se detallará el *movimiento de brazos*. Este estudio analizará como mover los brazos hacia delante y atrás, imitando el reparto de peso para equilibrarnos cuando nosotros andamos. Para hacerlo, se moverá la articulación q_1 de los brazos.

Tras definir todos los subconjuntos, se pueden empezar a analizarse uno por uno.

4.2.3. Subconjunto del *movimiento de avance*

El primer conjunto a analizar será el *movimiento de avance*. Para ello, basta con analizar sólo una de las dos piernas, pudiendo utilizar los resultados obtenidos para la otra. Como observación, hay que tener en cuenta el sentido de giro de cada motor, considerándose positivo al desplazarse en sentido antihorario y negativo en el opuesto. Este convenio será utilizado para todos los cálculos que se indique desde este momento, no siendo necesario citarlo constantemente.

En el análisis de este movimiento hay que tener en cuenta los elementos que participan, correspondiéndose con ciertas partes del robot. Como ejes de giro, se encuentran dos servomotores situados en la parte alta de la pierna y el tobillo. Estos grados de libertad corresponden a los elementos 3 y 6 según la notación vista en el apartado 2.2.1 *Descripción de la pierna*. Como se verá, otro elemento importante es la distancia que separa a ambos ejes. Esta distancia se corresponde con la longitud de la pierna en posición erguida, que es de aproximadamente 165,5mm. En ocasiones, se puede decidir contraer en cierta medida la longitud de la pierna para bajar el centro de gravedad, encogiendo parcialmente la rodilla del robot. En ese caso, la longitud de este elemento pude variar. A modo genérico, los cálculos se realizarán empleando el parámetro L que hará referencia a esta distancia. Para no aumentar en exceso el número de imágenes ni ecuaciones, se usará únicamente para el análisis la forma erguida de la pierna, siendo igual para un estado flexionado. Por último, existen otros dos ejes perpendiculares al plano de perfil, formando entre ambos ejes la articulación de la rodilla. Para simplificar el cálculo, estos ejes no se podrán mover en este subconjunto, quedando fuera del análisis.

Para ver como varía el giro de cada eje, se introduce el parámetro maestro para este subconjunto. Al ser el movimiento de avance, interesa definir una longitud horizontal referida desde el eje vertical que indique cuanto se desplaza el tobillo. A esta longitud se llamará *Avance* o simplemente A. Y queda definida de tal forma como se ve en la figura 4.7.

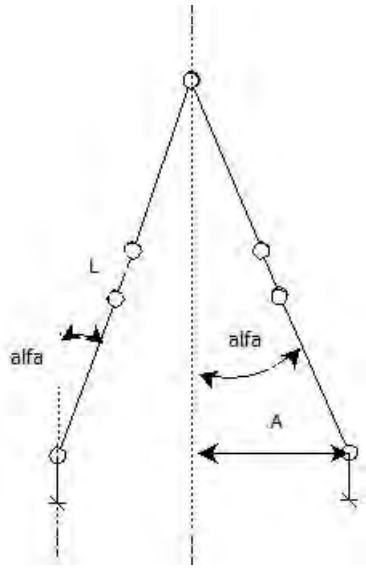


Figura 4.7: Definición del movimiento de avance. Vista Lateral.

Como se ve en la figura 4.7, la pierna en ese conjunto se asemeja a un péndulo. Para indicar el giro de los dos ejes se usará las varibles q_3 y q_6. A simple vista se puede ver que q_3 es el encargado de colocar al tobillo en la posición correspondiente y que q_6 se encarga de la orientación del pie en el plano. Ambos giros están referidos a la posición de inicio, siendo, en posición erguida, el eje vertical.

Si se analiza en detalle el subconjunto, se pude obtener la cinemática directa de un modo muy sencillo.

$$X = \sin(q_3)L \tag{4.1}$$

$$Y = \cos(q_3)L \tag{4.2}$$

$$\theta = q_3 + q_6 \tag{4.3}$$

Como condición de contorno, se pide que la planta del pie esté paralela al suelo en todo momento. La manera de realizar esto es que la suma de todos los ángulos hasta ese extremo sea cero. Para ello, el giro del tobillo a ha contraponerse al giro de la cadera, según la fórmula 4.3 se puede obtener 4.4.

$$q_3 + q_6 = 0 \tag{4.4}$$

$$q_3 = -q_6 = \alpha \tag{4.5}$$

Al tener ambos ejes el mismo giro pero de sentido opuesto, se definirá con el símbolo α para agrupar a ambos, como aparece en 4.5. Al imponer esto, se puede obtener α en función de A sustituyendo X por A en 4.1. Este ángulo, también quedará referido a la posición inicial.

$$\sin\alpha = \frac{A}{L} \tag{4.6}$$

Y al despejar α se obtiene finalmente la relación 4.7.

$$\alpha = \arcsin(\frac{A}{L}) \tag{4.7}$$

Como se indicó antes, α depende exclusivamente de la amplitud A y de la longitud de la pierna al estar su posición inicial L. Al ser L invariante en el tiempo, se puede considerar como una constante, pudiendo decir que α sólo está en función de A, es decir $\alpha(A)$.

En el caso de analizar la otra pierna, se obtendrían los mismo resultados excepto por el sentido de giro del motor. Al presentar simetría, y no cambiar el sentido de giro, si se indicase una avance positivo, los motores seguirían el angulo indicado por la misma fórmula, desplazando el tobillo en sentido opuesto. Para solucionar esto basta con cambiar el signo a α o A para poder usarla sin inconvenientes.

De este modo, según la fórmula 4.7, se puede empleando una única variable para controlar fácilmente el avance de la pierna durante el resto del proyecto.

4.2.4. Subconjunto del *movimiento de rodilla*

El siguiente análisis se procederá al estudio del subconjunto *movimiento de rodilla*. Como antes se indicó, este movimiento será el encargado de desplazar el tobillo verticalmente. Al igual que en el análisis anterior, sólo se estudiará la pierna derecha, teniendo una relación similar la izquierda salvo en el sentido de giro de los motores.

Para este estudio, es necesario conocer que elementos se incluyen y sobre que plano hay que analizarlos. Al imitar el movimiento humano, la rodilla ha de encogerse hacia adelante, esto da una pista que hay que usar el plano de perfil ya que en el se visualiza mejor el movimiento. Sobre este plano existen cuatro grados de libertad perpendiculares a él. Estos son los dos situados en

la rodilla, otro en el tobillo y, por último, el eje situado en la parte alta de la pierna. Siguiendo el orden ahora citada y apoyándose en lo explicado en el apartado 2.2.1 *Descripción de la pierna*, estos ejes se corresponden con los grados de libertad q_3, q_4, q_5 y q_6. Además de estos ejes, existen elementos que los unen, como los eslabones tres, cuatro y cinco.

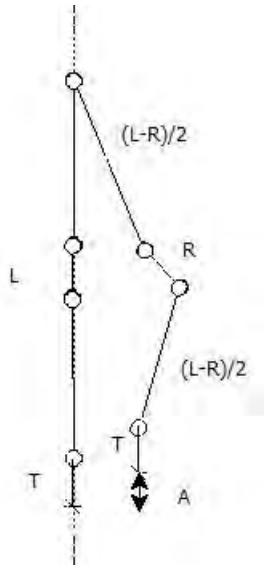


Figura 4.8: Presentación del movimiento de Rodilla. Vista lateral.

Al igual que sucede en 4.2.3, se considera la longitud entre los ejes extremo como L, pudiendo ser la correspondiente a la distancia en posición erguida o en otra con las rodillas parcialmente encogidas. También es necesario distinguir la distancia entre los ejes de la rodilla R, ya que será de mucha utilidad. Esta distancia R es de un total de 27,5 mm. Esta distancia no solo está presente en entre estos dos ejes, si no que por motivos de diseño al repetir el tipo de pieza utilizado, si no que también lo está entre los ejes q_2 y q_3, que son los ejes de la parte alta de la pierna; q_4 y q_5, que corresponde con la rodilla y q_6 q_7, correspondientes a la articulación del tobillo.

Como se puede ver en la figura 4.8, al tener cuatro grados de libertad en un plano, existen casi infinitas configuraciones para posicionar y orientar el extremo por ser un sistema redundante. Para ello, se ha de recurrir a diferentes ecuaciones de que delimiten el espacio solución.

Se ha dicho antes que para este movimiento solo interesa que el tobillo se desplace verticalmente. Esto restringe el área total de las posiciones posibles del tobillo a un línea recta. Para conseguir esto hay que imponer una condición de contorno sobre la ecuación del eje X de la cinemática inversa para que el desplazamiento en ese eje sea 0.

$$\sin(q_3) \cdot l_3 + \sin(q_4 + q_3) \cdot l_4 + \sin(q_5 + q_4 + q_3) \cdot l_5 = 0 \tag{4.8}$$

Se sabe que la distancia de el eslabón 4 es R. Por el diseño del robot, los eslabones l_3 y l_5 tienen la misma longitud $l^{'}$, la cual se corresponde con $\dfrac{L-R}{2}$. Sustituyendo queda.

$$\sin(q_3) \cdot l^{'} + \sin(q_4 + q_3) \cdot R + \sin(q_5 + q_4 + q_3) \cdot l^{'} = 0 \tag{4.9}$$

Por otro lado, se desea imponer que el eslabón de la rodilla sea siempre normal al plano horizontal. Del mismo modo que la ecuación 4.5, es necesario que los valores de q_3 y q_4 se compensen. Por esto, se puede escribir lo siguiente.

$$q_3 = -q_4 = \alpha \tag{4.10}$$

Al incluir 4.10 en 4.9 y sabiendo que $q_3 + q_4 = 0$ se obtiene

$$\sin(\alpha) \cdot l^{'} + \sin(q_5) \cdot l^{'} = 0$$

Por último, se intenta despejar el valor de q_5 en función de α.

$$\sin(\alpha) \cdot l^{'} = -\sin(q_5) \cdot l^{'}$$

$$\sin(\alpha) = -\sin(q_5)$$

$$q_5 = -\alpha \tag{4.11}$$

Como se puede ver en las ecuaciones 4.11 y 4.10, sólo existe un único parámetro que controle la posición del extremo al usar las antes mencionadas condiciones de contorno. Esto es una ventaja al sólo tener que preocuparse por una variable en lugar de tres.

Tras esto, hay que imponer una condición de contorno para asegurarse de que la planta del pie quede siempre paralela al suelo. Para ello es suficiente imponer que el angulo total girado por los ejes sea cero.

$$q_3 + q_4 + q_5 + q_6 = 0 \tag{4.12}$$

Recurriendo a 4.11 y 4.10 se puede ver que

$$\alpha - \alpha - \alpha + q_6 = 0$$

$$-\alpha + q_6 = 0$$

Lo que significa que

$$q_6 = \alpha \tag{4.13}$$

Con esta última ecuación se hace depender a la cuarta variable de α. Al tener agrupadas las cuatro variables en una sola se verá que el control de este movimiento se simplifica notablemente. Por ello, como se ve en la figura 4.9 y en la ecuación 4.14, el movimiento de la rodilla queda controlado por un solo parámetro.

$$\alpha = q_3 = q_6 = -q_4 = -q_5 \tag{4.14}$$

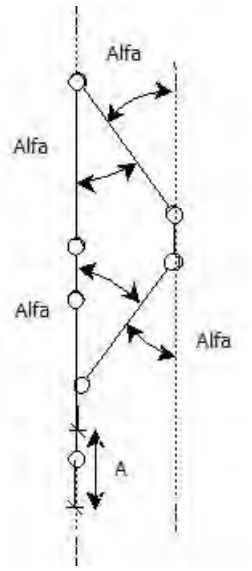


Figura 4.9: Definición del movimiento de Rodilla. Vista lateral.

Una vez que se ha conseguido hacer depender todos los desplazamientos en función de uno de ellos, se puede intentar plantear dejar estos grados de libertad dependientes de un parámetro.

Este parámetro será el que defina qué altura ha de levantar del suelo el pie del robot, pudiendo parametrizar el *movimiento de rodilla*. Para ello, sin olvidar las ecuaciones restrictivas que se han impuesto, se puede analizar al conjunto en función de este parámetro *Amplitud o A*. Con la idea de facilitar la comprensión y visualización de este análisis es necesario apoyarse en la figura 4.9.

El planteamiento para relacionar α con A consiste en relacionar la longitud total de la pierna en la posición de reposo con la distancia en la que se deba encoger lo suficiente la pierna para colocar el tobillo en esa posición. Para ello se pude realiza el cálculo con la ecuación 4.19, resultante de sustituir A en la ecuación de la cinemática directa del eje Y, o usando un método gráfico.

Método gráfico

En primer lugar, se comenzara por el método gráfico. Para entender lo siguiente, es necesario señal que el segmento correspondiente a l_4 con la longitud R siempre estará en vertical gracias a las condiciones antes impuestas. Esto quiere decir que, siempre que se mida verticalmente la distancia entre los extremos del eslabón cuatro dará como resultado R. A modo de hacer más sencillos los cálculos y la fórmula final, se analizará el mecanismo recolocando los elementos para que su disposición sea alineada, respetando por supuesto el giro que provoca α. Para visualizar esto, es necesario apoyarse en la figura 4.10.

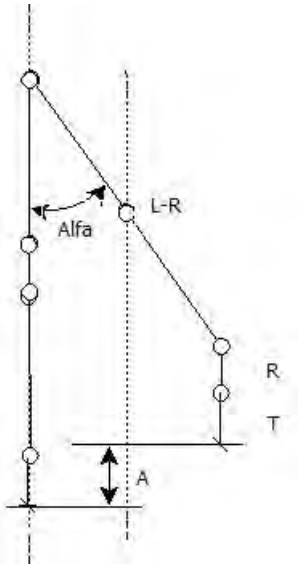


Figura 4.10: Recolocación de los elementos para analizar el movimiento de rodilla.

Este análisis sólo se basa en la disposición de cada elemento medida sobre el eje vertical. Esto permite que se puedan recolocar y girar ciertos elementos para la mejor visualización y análisis, siempre que no se varíe su distancia vertical al reorientarlos. Por esto se pueden hacer los cambios que se muestran a continuación.

Como se ve en la figura, se ha colocado el eslabón cinco a continuación del tercero. Estos quedan alineados al suponer que en la unión de ambos extremos se encuentran los grados de libertad cuarto y sexto, anulándose entre sí. Del mismo modo, se reubica al quinto eje al extremo de quinto eslabón. por último, sobre este mismo eje se apoya el cuarto eslabón R.

Otra forma de ver esta reubicación es hacer una simetría de la parte inferior de la pierna sobre el eslabón l_4 o R. Tras espejar el quinto eslabón, se pueden cambiar el orden en el que aparecen el cuarto y quinto eslabón, de tal forma que queden alineados el tercer y quinto eslabón, y en la punta de estos quede el eslabón cuatro.

Esta recolocación de los elementos responde a la necesidad de analizar el mecanismo como si fuera un péndulo, de modo similar al apartado del *movimiento de avance*. Al hacer esto, se facilitan los cálculos, obteniendo sólo unos pocos parámetros.

Como se ha dicho anteriormente, este análisis reside en estudiar el péndulo generado en la posición más baja y en cualquier otra posición, siendo el parámetro A la diferencia vertical entre ambas. Si se observa la figura 4.9, se pueden cuantificar estas dos alturas.

$$L' = L - A \tag{4.15}$$

Esta altura L' corresponde a la diferencia vertical entre la longitud total de la pierna en posición erguida menos la distancia que se quiera encoger. Del mismo modo, la ecuación 4.16 responde a la distancia vertical que se genera al girar el péndulo un ángulo α genérico junto con el eslabón de la rodilla.

$$L' = (L - R)\cos(\alpha) + R \tag{4.16}$$

Concepto de $L - R$

En este punto es importante señalar que significa gráficamente la resta de los parámetros $L - R$. Esta resta, que se verá con frecuencia a lo largo de los estudios de los subconjuntos, responde a la valor real que la pierna puede doblarse. Como se ha dicho, el eslabón cuarto o R siempre está vertical, sin importar cual sea el valor de α. Por lo tanto, si se separa la longitud L, que representa a la distancia total de la pierna, de R, que corresponde a la parte de la pierna que no puede rotar, se obtiene la parte real que puede variar su longitud vertical o que puede rotar $L - R$. Tras aclarar esto se puede continuar con el desarrollo.

Como se puede ver, en ambos caso L' han de ser idénticas pudiendo igualarlas para obtener la ecuación 4.17.

$$L - A = (L - R)\cos(\alpha) + R \tag{4.17}$$

Una vez echo esto, están relacionadas en la misma ecuación α y A, procediendo a despejar la primera de ellas en función de la otra.

$$\begin{aligned} \cos(\alpha)(L - R) &= L - R - A \\ \cos(\alpha) &= \frac{(L - R) - A}{(L - R)} \\ \alpha &= \arccos\left(\frac{(L - R) - A}{(L - R)}\right) \end{aligned} \tag{4.18}$$

Método matemático

Por otro lado, se pude obtener el mismo resultado si despejamos la ecuación 4.19 relativa a la cinemática directa del eje Y del subconjunto igualándolo a A.

$$A = L - (l_3\cos(q_3) + l_4\cos(q_4 + q_3) + l_5\cos(q_5 + q_4 + q3)) \tag{4.19}$$

Aplicando todas los cálculos antes realizados se obtiene

$$A = L - (l^{'}\cos(\alpha) + R\cos(-\alpha + \alpha) + l^{'}\cos(-\alpha - \alpha + \alpha))$$

$$A = L - (l^{'}\cos(\alpha) + R\cos(0) + l^{'}\cos(-\alpha))$$

$$A = L - R - (l^{'}\cos(\alpha) + l^{'}\cos(-\alpha))$$

Si se considera $\cos(\alpha) = \cos(-\alpha)$ se obtiene finalmente

$$A = L - R - 2(\frac{L-R}{2})cos(\alpha) \tag{4.20}$$

Como se puede ver, esta última ecuación es idéntica a 4.17. Esto asegura que ambas formas de calcular esta relación dan el mismo resultado. Por último basta con despejar de nuevo α para poder obtener lo mismo que en 4.18.

$$\cos(\alpha)(L-R) = L - R - A$$

$$\cos(\alpha) = \frac{(L-R) - A}{(L-R)}$$

Por último, para concluir

$$\alpha = \arccos\left(\frac{(L-R) - A}{(L-R)}\right) \tag{4.21}$$

Como se puede ver en el desarrollo anterior, la ecuación 4.21 relaciona α con el desplazamiento vertical del mismo modo que la ecuación 4.18. Esta ecuación se puede analizar de un modo que cada parámetro corresponda con un elemento geométrico del modelo real. Se uno se fija en ella, esta presente la relación

$$\frac{(L-R) - A}{(L-R)} \tag{4.22}$$

En el nominador aparece la nueva longitud de la parte móvil de la pierna, $L - R$ al restar a $L - R$ la cantidad de acortamiento A. Por el contrario, en el denominador está de forma constante el valor $L - R$, representado a la longitud máxima de acortamiento. Basándose en esto, se puede decir que α depende exclusivamente de la relación entre la pierna en posición encogida y en posición estirada.

Por último, hay que tener en cuenta que $\cos(\alpha) = cos(-\alpha)$, por lo que existen dos soluciones reales α para la ecuación 4.21. Un único valor de A provocará las dos soluciones de α pero con signo opuesto. En este proyecto se ha de conseguir que el robot ande de una forma similar a como lo hacen los humanos, doblando la rodilla hacia adelante. Pese a que ambas soluciones son posibles, solo una de ella provoca el efecto antes citado. Por esto es necesario recurrir a una última restricción, que delimite una de las dos opciones. Como se puede en 4.11, necesitamos que la rodilla doble en el sentido de avance del robot, solo pudiendo obtener valores positivos en α. Dicho esto, la última ecuación de contorno quedará definida como aparece en 4.23.

$$\alpha \geq 0 \tag{4.23}$$

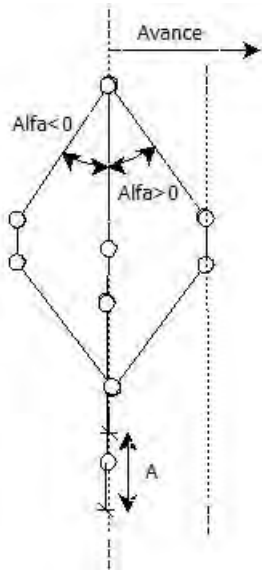


Figura 4.11: Disposición del conjunto de rodilla *Codo Arriba* y *Codo Abajo*.

4.2.5. Subconjunto del *movimiento de balanceo*

El *movimiento de balanceo* es el encargado de repartir el peso del robot entre una pierna y otra. Este movimiento es vital para poder desplazar correctamente el robot sin que arrastre los pies. Como se ha explicado anteriormente, existen dos versiones de la pieza de que une cada pierna con la cadera. Dependiendo de la geometría de esta se pueden obtener cadenas cinemáticas diferentes.

En este estudio no solo interesa la geometría de los elementos si no que también importa como se quiera definir el movimiento en función de ciertas condiciones de contorno. Estas ecuaciones extra permitirán obtener diferentes valores, en función de las diferentes formas de posicionar los elementos. Además, se pueden combinar ambas opciones al poder analizar una de las dos versiones imponiendo diferentes restricciones. Esto provocará las diferentes cadenas cinemáticas y movimientos que se analizarán en este apartado.

A diferencia del resto de subdivisiones del robot, en los siguientes apartados se analizarán en conjunto ambas piernas. Para relacionarlas a ambas también se analiza la pieza que las une, que es la cadera. Para refrescar esto, es aconsejable volver a mirar la figura 2.7 donde se puede ver una comparativa entre las configuraciones de las que ahora se habla. Es importante citar según lo visto en la *Descripción general* 2.2.1 que, la distancia entre los ejes q_2 de cada pierna varía en función de la versión del robot. Del mismo modo, la distancia horizontal que separa a los ejes q_2 y q_7 de una misma pierna también se ve afectada. Para denominar a ambas versiones de una forma intuitiva, desde ahora se le llamará *Versión de cadera ancha* a la configuración que ofrece más distancia entre los ejes q_2, denominando a la otra como *Versión de cadera estrecha*.

Versión de cadera ancha

Esta versión de montaje del robot ofrece una distancia entre los ejes q_2 de cada pierna de 106 mm. Esto provoca además que los ejes q_2 y q_7 de cada una de las piernas no queden alineadas, quedando separadas horizontalmente 20 mm, que será llamada desde ahora distancia horizontal entre ejes d_h. Por otro lado, de forma invariante a que versión se utilice, la distancia total vertical medida entre los mismos ejes es de 110.5 mm en posición estirada. Con estos datos, si los representamos gráficamente sobre el plano de alzado se obtiene una figura que se asemeja a un trapecio que se apoya sobre la base menor, como se muestra en la figura 4.12. En cada vértice del trapecio se encuentran los cuatro ejes participantes, que son q_2 y q_7 de cada una de las piernas.

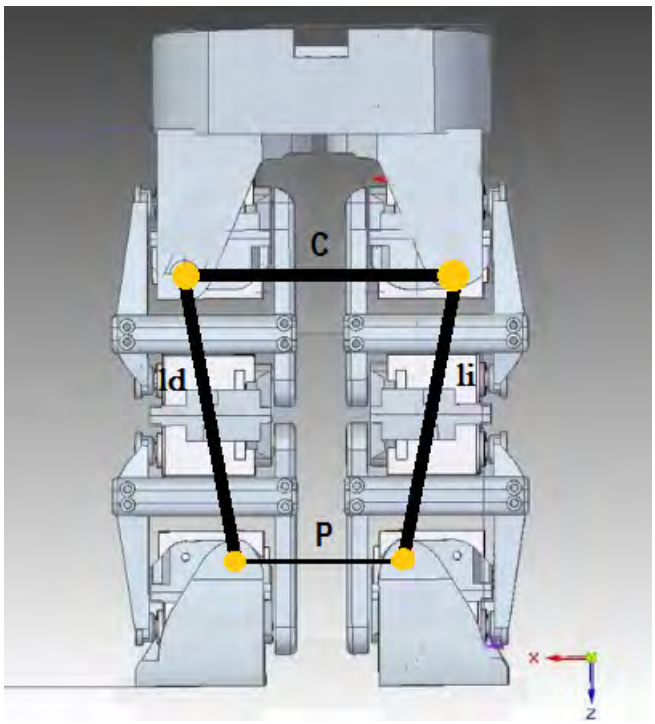


Figura 4.12: Descripción del trapecio formado por la versión ancha.

Este trapecio será de gran importancia durante el estudio de los movimientos de esta versión al realizar todos los cálculos gráficamente sobre el. De modo diferente al resto de estudios, el trapecio constituye una cadena cinemática cerrada, al no tener un extremo libre. Como se verá desde ahora en adelante, los ejes q_7 se considerarán totalmente fijos en el espacio, pudiendo rotar únicamente sobre sí mismos. Esto responde a la necesidad de calcular esta cadena cinemática de forma de que genere un mecanismo.

En este caso, el trapecio originado tiene cuatro eslabones, si se considera a uno de ellos como tal a la distancia que une los dos ejes q_7, que se articulan mediante ejes rotativos. Para simplificar la notación y ser más precisos, se detallan a continuación.

Eslabones del trapecio

Eslabón cadera C.

Este eslabón está situado en la parte superior del trapecio teniendo en los extremos a los ejes q_2 de cada pierna. Por motivos geométricos, al no existir ningún elemento en el plano de alzado que pueda variar esto, la distancia entre ambos ejes siempre será de 106 mm, quedando definidas las dimensiones del eslabón.

Eslabón distancia entre tobillos P

Esta distancia corresponde con la base menor del trapecio. Este eslabón se considera totalmente inmóvil, sin poder desplazar ni rotar su posición. A los extremos de él se encuentran ambos ejes q_7, que se separan una distancia de 81 mm. Pese a poder variar esta distancia al rotar los ejes q_7 o al acotar la pierna girando los ejes q_3, q_4, q_5 y q_6, se impone como condición del estudio que no se pueda variar de ningún modo. Esta condición emula que ambos pies no se separen del suelo y que el movimiento final esté destinado a la cadera.

Eslabón pierna derecha l_d

Este eslabón queda definido por la unión en línea recta de los ejes q_2 y q_7 de la pierna derecha, desde ahora q_{d2} y q_{d7}. La distancia de este eslabón es $\sqrt{l'^2 + d_h^2}$, siendo l' la distancia vertical entre los ejes q_{d2} y q_{d7} en posición de reposo. Hay que señalar que l' no es igual a L, al ser otra distancia medida sobre ejes diferentes. No obstante, por motivos

de diseño, estos parámetros están relacionados por $l^{'} = L - 2R$. Esta distancia $l^{'}$ si que puede variar al poder definir como condición de contorno en el caso que se quiera al poder articular la rodilla de forma similar a lo visto en el apartado correspondiente a *movimiento de rodilla*. En cualquier caso, en posición estirada este $l^{'}$ valor corresponde $110.5mm$, definiendo la longitud total de 113.87 mm según la fórmula $\sqrt{l'^2 + d_h^2}$.

Eslabón pierna derecha l_i

De modo similar al anterior, este eslabón esta definido por los ejes q_2 y q_7 de la pierna izquierda, q_{i2} y q_{i7}. Por ser idéntico al derecho, cumple las mimas condiciones y características que l_d.

Como se usará frecuentemente en los siguientes cálculos es aconsejable definir también la distancia total entre cualquier q_2 y q_7 como l. Esta longitud corresponde a la fórmula

$$l = \sqrt{l'^2 + d_h^2} \tag{4.24}$$

que ya apareció anteriormente. Se puede ver fácilmente que los tres parámetros corresponden con los lados de un triángulo con un ángulo recto. En la ecuación 4.24 se pude ver que l depende de los otros dos parámetros. Por motivos geométricos en el diseño del robot, d_h es siempre una distancia constante, al estar definida como la distancia medida sobre la recta normal a la que une a los ejes q_2 y q_{q7} y no existir ningún elemento que pueda variar esta distancia horizontal en la cadena entre ambos ejes de rotación. Por esto l variará únicamente en función de la variable l' por ser d_h invariante en el tiempo. Por otro lado, el ángulo φ generado entre los elementos l y l' también varía en función de como lo haga el segundo. Si se aplica trigonometría, se pueden relacionar los términos con la ecuación 4.25.

$$\varphi = \arctan \frac{d_h}{l'} \tag{4.25}$$

Del mismo modo que si se sustituye 4.24 en 4.25 se puede sacar la relación

$$l' = l \cos \varphi \tag{4.26}$$

A modo de señalar dos formas diferentes para mover la cadera, se ha optado por generar dos movimientos visiblemente diferentes. Uno de ellos solo tendrá un desplazamiento lateral, mientras que el otro, además de este desplazamiento, añadirá un pequeño giro relativo que permitirá rotar el torso del robot respecto al suelo.

Giro de cadera con desplazamiento Tras definir el paralepípedo articulado en el cual se basará el estudio, se pude empezar a definir los primeros movimientos del mismo. En este primer estudio se analizará el giro de cada uno de los elementos presentes al imponer que el punto medio de la cadera C_o se desplace una distancia en horizontal medida desde la posición de reposo. Esta distancia se denotará por el símbolo *desp*, y se puede ver en la figura 4.13.

Por ser un trapecio con las bases de diferente longitud, el desplazamiento en horizontal del punto medio de la cadera provocará también una rotación en el eslabón C. Como la base del trapecio con mayor longitud está situada en la parte alta y móvil del mecanismo, un desplazamiento hacia la derecha del C_o sobre el eje horizontal provocará que el eslabón rote con un sentido de giro horario.

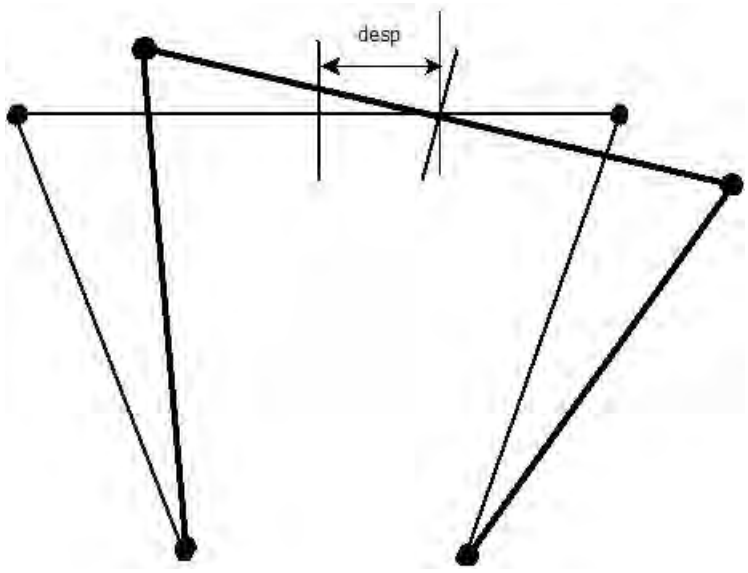


Figura 4.13: Mecanismo tras el desplazamiento.

Si se supone que una porcentaje significativo del peso del robot está apoyado sobre la cadera C, al sumar los efectos del movimiento del desplazamiento lineal y de rotación del centro del eslabón superior, se puede desplazar el centro de masas del robot para poder balancearlo.

Para poder realizar esto es necesario relacionar el parámetro de desplazamiento *desp* con el giro que se provoca en cada uno de los cuatro grados de libertad. Para analizar esto es suficiente relacionar el desplazamiento vertical y horizontal de los dos vértices superiores, A y B. Para facilitar el estudio, se girarán q_{d7} y q_{i7} el mismo ángulo α y se verá en que punto del plano queda C_o.

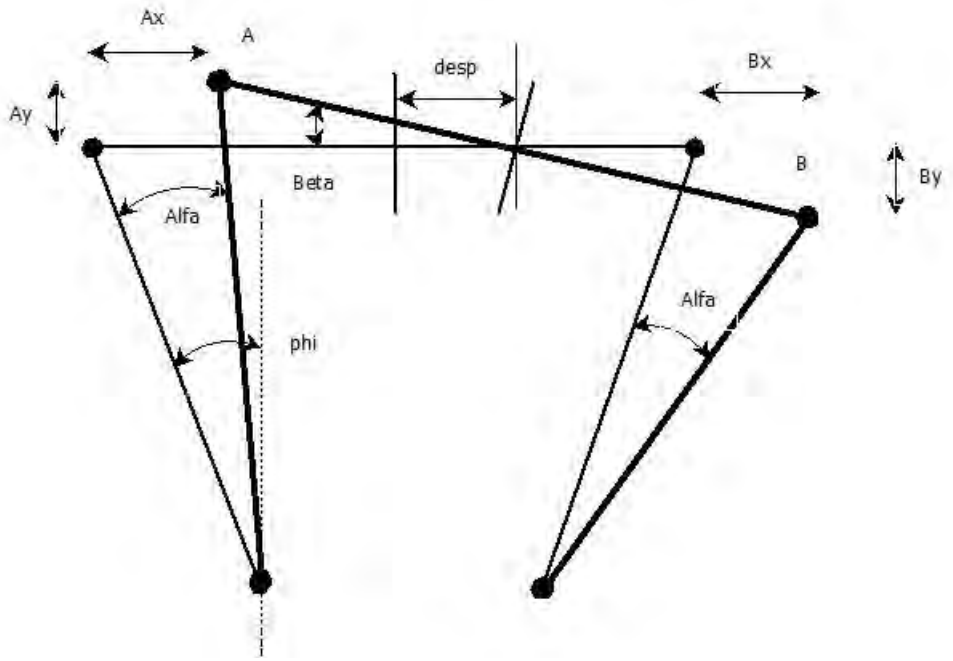


Figura 4.14: Definición de los elementos partícipes en es movimiento de giro de cadera con desplazamiento.

Para visualizar mejor el movimiento del robot se va a analizar el trapecio desde la parte trasera, quedando a la derecha del esquema la pierna derecha y viceversa. Además se puede ver mejor la rotación de los servomotores al visualizar directamente el eje de salida de cada uno de

ellos.

En las siguientes cuatro fórmulas se puede ver el movimiento relativo de los vértices de forma matemática, ubicando cada origen de referencias en el q_7 correspondiente a cada pierna. El convenio de giro de los ángulos en este caso es diferente hasta lo ahora usado, aumentando en sentido horario. Este criterio simplemente se ha utilizado para ahorrar tener que poner signos negativos en la mayoría de los ángulos al tener un desplazamiento positivo hacia la derecha.

$$A_x = l\sin(\varphi - \alpha) - l\sin\varphi \tag{4.27}$$

$$B_x = l\sin(\varphi + \alpha) - l\sin\varphi \tag{4.28}$$

$$A_y = l\cos(\varphi - \alpha) - l\cos\varphi \tag{4.29}$$

$$B_y = l\cos(\varphi + \alpha) - l\cos\varphi \tag{4.30}$$

Para saber cual es la distancia en vertical que realmente se han alejado los vértices A y B, basta con restar A_y y B_y.

$$A_y - B_y = l[(\cos(\varphi - \alpha) - \cos(\varphi)) - (\cos(\varphi - \alpha) - \cos(\varphi))] \tag{4.31}$$

$$A_y - B_y = l[\cos(\varphi - \alpha) - \cos(\varphi - \alpha)]$$

Si se usa la relación $\cos(\varphi - \alpha) = \cos\varphi\cos\alpha + \sin\varphi\sin\alpha$ se obtiene

$$= l\cos(\varphi - \alpha) = \cos\varphi\cos\alpha + \sin\varphi\sin\alpha - \cos(\varphi - \alpha) = \cos\varphi\cos\alpha + \sin\varphi\sin\alpha =$$

$$= 2l\sin\varphi\sin\alpha =$$

$$= 2\frac{l'}{\cos\varphi}\sin\varphi\sin\alpha =$$

$$= 2l'\tan\varphi\sin\alpha =$$

$$= 2l'\frac{d_h}{l'}\sin\alpha$$

Por lo que finalmente se obtiene 4.32

$$A_y - B_y = 2d_h\sin\alpha \tag{4.32}$$

Con el dato de 4.32 se puede hallar fácilmente el ángulo girado por la cadera β si se construye el siguiente triángulo.

$$\beta = \arcsin\frac{2d_h\sin\alpha}{C} \tag{4.33}$$

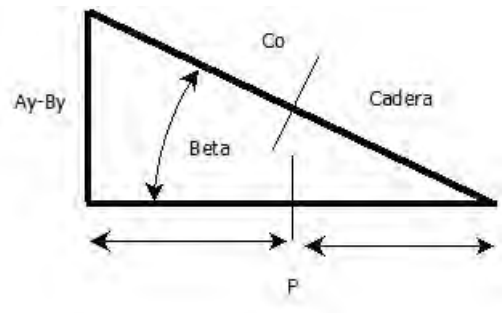


Figura 4.15: Detalle de como obtener β.

Como se ve en 4.33, se ha conseguido hacer depender el giro de la cadera β de al ángulo α que se gire, además de relacionarlos con los parámetros geométricos de la cadera y la separación entre ejes.

En este punto es necesario obtener una relación directa entre el desplazamiento de la cadera *desp* y el valor de α. Para ello, volviéndose a apoyar en el triángulo de la figura 4.15, se ha de analizar el desplazamiento sobre el eje horizontal. Aplicando el *teorema de Thales*, el punto medio de la cadera C_0 se proyecta con el punto medio del lado horizontal del triángulo P, que es el que hay que analizar para ver el desplazamiento en ese eje.

Usando este triángulo y apoyándose en el punto A_x se puede plantear la ecuación de desplazamiento de la forma *desp = posición final de P - posición de origen de P.*

$$desp = (A_x + \frac{C}{2}\cos\beta) - \frac{C}{2} \tag{4.34}$$

$$desp = l[\sin(\varphi - \alpha) - \sin\varphi] + (\cos\beta - 1)\frac{C}{2}$$

Usando la propiedad $\sin(\varphi - \alpha) = \sin\varphi\cos\alpha - \cos\varphi\sin\alpha$

$$\frac{desp}{l} = \cos\varphi\sin\alpha - \sin\varphi\cos\alpha + \sin\varphi + \frac{(\cos\beta - 1)C}{2l}$$

Se intenta despejar uno de los términos de α en función del resto

$$\sin\alpha\cos\varphi = \sin\varphi\cos\alpha - \sin\varphi + \frac{desp}{l} - \frac{(\cos\beta - 1)C}{2l}$$

$$\sin\alpha = \tan\varphi\cos\alpha - \tan\varphi + \frac{desp}{l\cos\varphi} - \frac{(\cos\beta - 1)C}{2l\cos\varphi}$$

Se sustituye en esta ecuación lo visto en 4.25 y finalmente se obtiene

$$\alpha = \arcsin[\frac{d_h}{l'}(\cos\alpha - 1) + \frac{desp}{l'} - \frac{(\cos\beta - 1)C}{2l'}] \tag{4.35}$$

Esta ecuación relaciona finalmente el giro que se ha de producir en los ejes q_7 con el desplazamiento horizontal que se requiera. Se puede apreciar que esta ecuación esta dividida en tres términos que tienen un significado geométrico que está relacionado con los elementos del robot. En el tercero de estos términos aparece el parámetro β. Como se ha visto en la ecuación 4.33, esta variable depende también de α. Si se incluye esto en la ecuación, el tercer término quedará como

$$\frac{[\cos[\arcsin(\frac{2d\sin\alpha}{C})] - 1]C}{2l'}$$

Como se pude ver en la ecuación 4.35 el valor final de α depende de si mismo al estar presente también en dos de los tres términos participantes. Esto ocasiona que para resolver esta ecuación es necesario iterar.

Para solucionar la ecuación 4.35 es necesario haber introducido el valor requerido del desplazamiento junto con los parámetros que responden con la geometría del robot. Tras esto, es necesario dar un valor inicial a α. Este valor puede ser cualquier número real, pero si analizamos en detalle la ecuación se puede ver que si se sustituye α por 0, tanto el primer como el tercer término se anulan, solo afectando al valor final el segundo término. Por esta razón, es conveniente comenzar iterando con un valor de α igual a cero. Por ser una función que depende en mayor medida de relaciones trigonométricas, la solución no solo converge en cualquier caso real si no que se puede hallar este valor en unas pocas iteraciones si se parte del valor inicial cero.

Esto, aunque no sea tan rápido como una sustitución única de valores, requiere poco esfuerzo y tiempo para obtener el resultado final.

Por último, es necesario conocer cual es el giro en los ejes q_{d2} y q_{i2}. Como se puede ver en la figura 4.14, es muy sencillo relacionar el giro en los ejes con los datos hallados hasta ahora. Por esto, la relación final de este giro θ se corresponde con

$$\theta = -\alpha + \beta \tag{4.36}$$

Por lo tanto, haciendo recopilación de lo visto en este apartado, el giro que hay que realizar en los motores está reflejado en las ecuaciones de α 4.35 y de θ 4.36.

$$q_{d7} = q_{i7} = \alpha \tag{4.37}$$

$$q_{d2} = q_{i2} = \theta \tag{4.38}$$

Movimiento horizontal de cadera Otra forma de repartir el peso del robot en cada pierna con esta configuración puede ser moviendo la cadera horizontalmente sin que varíe su orientación o, dicho de otro modo, que la cadera siempre quede paralela al suelo sea cual sea la posición que ocupe. Como se ha dicho antes, el mecanismo tiene ciertas restricciones que lo hacen isoestático. Si se quiere tener una imposición más que obligue al mecanismo a mantener su orientación, es necesario generar un nuevo grado de libertad que permita al mecanismo adecuarse a esta nueva restricción. Para poder disponer de esto se va a emplear como grado de libertad añadido la variación de longitud de una pierna. Al poder estirar o encoger la pierna sí se puede mantener la orientación de la cadera al hacer que la altura medida verticalmente desde el suelo en una y otra pierna sea idéntica.

Por motivos obvios, al usar articulaciones rotativas con eslabones de longitud constante, la longitud de la pierna no puede aumentar más de su posición estirada. En ciertos casos, si se considera que el eslabón l_i es el que puedan variar su longitud pueden aparecer complicaciones como se muestra en el siguiente ejemplo. Si el eslabón l_d del trapecio queda en posición vertical, para poder hacer que la cadera siga estando en posición horizontal la barra l_i deberá tener una longitud total mayor de lo que la pierna puede llegar a alcanzar en posición estirada. Por este motivo no se puede tener únicamente un eslabón variable.

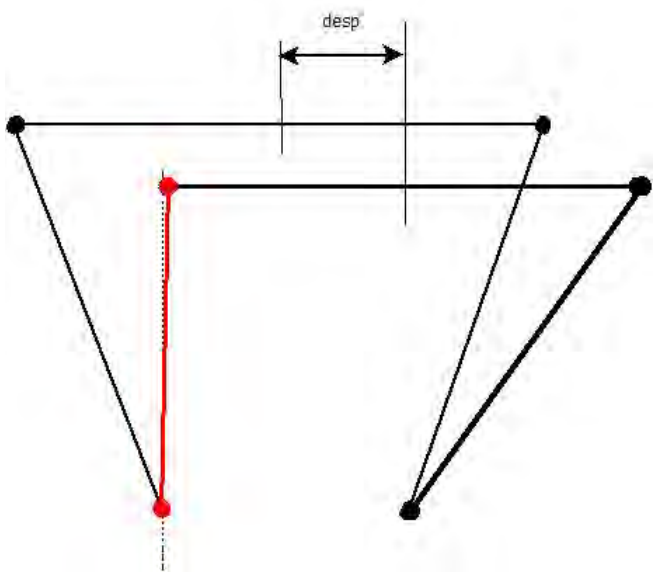
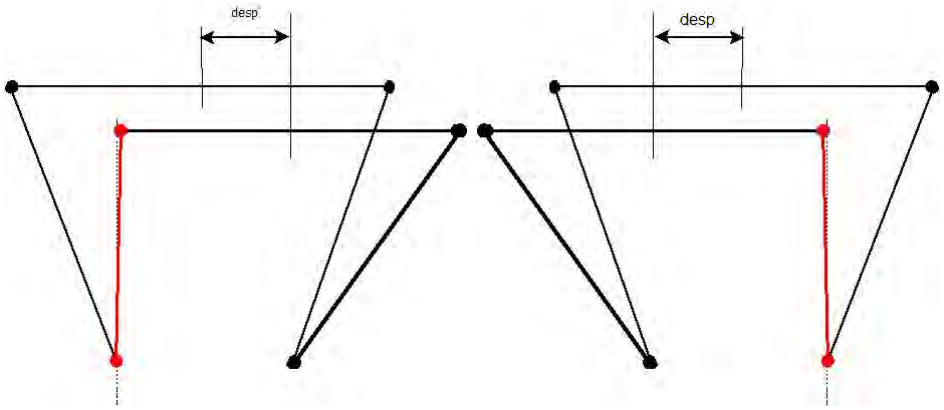


Figura 4.16: Necesidad de acortamiento de una barra.

Para solucionar este inconveniente es necesario recurrir a que ambos eslabones laterales l_d y l_i puede variar su longitud. Dependiendo de la dirección del desplazamiento uno u otro eslabón

será el que se encoja para adaptarse a la condición de contorno. Para no tener infinitas soluciones al meter un segundo grado de libertad, es necesario imponer que solo se puede encoger una de las dos piernas en una posición. De este modo podemos conseguir el objetivo propuesto.



(a) Desplazamiento a la derecha. (b) Desplazamiento a la izquierda.

Figura 4.17: Se acorta las dos piernas dependiendo de la dirección del movimiento.

Para no tener dos estudios idénticos para cada una de las dos piernas, se analizará a continuación un único estudio para un desplazamiento *desp* positivo, pudiendo ser extrapolable para la otra situación. Además es necesario citar que el sentido de giro que se considerará positivo para este análisis será el horario, por los mismos motivos descritos en el apartado anterior.

En primer lugar se mostrarán que elementos participan en este estudio en la figura 4.18. El la vista donde se representan estos elementos es la trasera, pudiendo observar los ejes de giro de los motores.

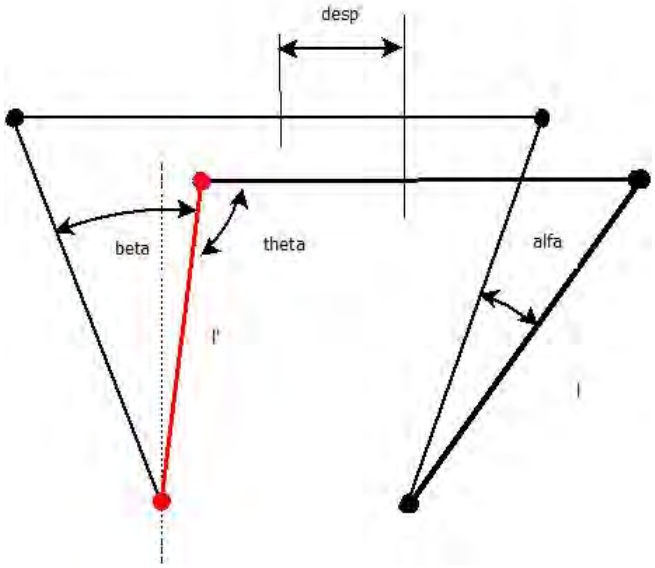


Figura 4.18: Presentación de parte de las variables presentes en el estudio Movimiento horizontal de cadera.

Como se puede ver en la figura, existen solo dos ángulo, α y β que definen el trapecio por estar presente la condición de la horizontalidad de la cadera. Esta restricción, como se ha visto anteriormente, implica que la orientación del extremo en cada pierna θ tenga el mismo valor que la horizontal, que se considera cero. Por lo tanto, la suma de los ángulos presentes en los eslabones l_d y l_i ha de ser nula

$$q_2 + q_7 = 0 \tag{4.39}$$

Esto implica que cada pierna está gobernada por un único angulo. Como convenio, α corresponde al ángulo que hace girar a la pierna estirada mientras que β lo hace con la pierna que tiene una longitud variable. En el caso de que el desplazamiento sea hacia la derecha, el eslabón l_d está asociado a α y l_i lo está a β. Si se une lo visto hasta ahora con la ecuación de cada pierna se puede obtener fácilmente el giro para cada grado de libertad.

$$q_{d7} = \alpha \tag{4.40}$$

$$q_{d2} = -\alpha \tag{4.41}$$

$$q_{i7} = \beta \tag{4.42}$$

$$q_{i2} = -\beta \tag{4.43}$$

Para analizar cual es el giro que se produce en α y β al posicionar la cadera a una distancia *desp* del punto de referencia se pueden dividir dichos ángulos en otros dos para simplificar los cálculos. Al sumar o restar estos dos pequeños ángulos se puede obtener de nuevo el resultado que se desea.

En primer lugar se empezará a analizar el mecanismo por la parte correspondiente a α. Para poder analizarlo de un modo más sencillo, se definirán los ángulos α' y α''. Como se puede ver en la figura 4.20, los tres ángulos mencionados hasta ahora tienen su referencia en la línea vertical medida desde el tobillo. Se puede ver de un modo muy intuitivo que la diferencia entre α'' y α' da como resultado α.

$$\alpha = \alpha'' - \alpha' \tag{4.44}$$

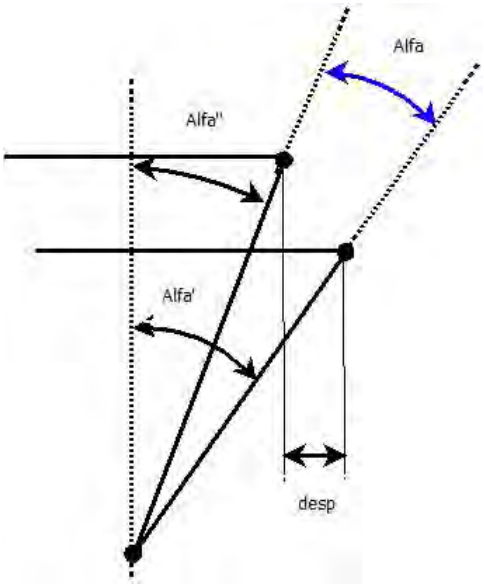


Figura 4.19: Detalle de la parte derecha del mecanismo.

El ángulo α' se puede definir como el ángulo entre la vertical y el eslabón l_d al separar el extremo del segundo una distancia horizontal d_h. Como se puede ver todos los elementos participes en esta definición son constantes, por lo que el valor de α' se puede considerar constante. Por otro lado, se puede definir al ángulo α'' como la apertura entre la vertical y el eslabón l_d al separar el extremo del segundo una distancia igual a la suma de d_h y *desp*. En este caso, todos los elementos de la definición con constantes salvo *desp*. Esto provoca que α'' quede en función de la variable que define el movimiento, cambiando el valor final en cada posición.

$$\alpha' = \arcsin \frac{d_h}{l_d} \tag{4.45}$$

$$\alpha'' = \arcsin \frac{d_h + desp}{l_d} \tag{4.46}$$

Si se recupera la definición de α definida previamente en 4.44, y se incluyen las definiciones de α'' y α', el ángulo que se busca queda definido totalmente.

$$\alpha = \alpha'' - \alpha' = \arcsin \frac{d_h + desp}{l_d} - \arcsin \frac{d_h}{l_d} \tag{4.47}$$

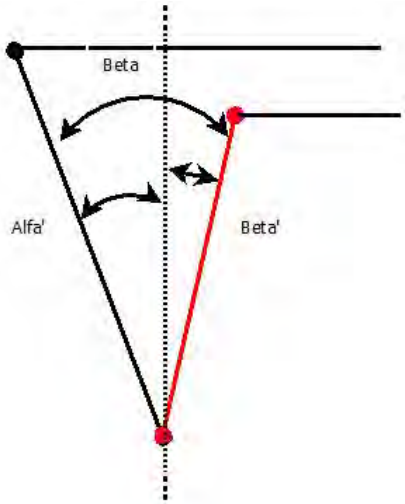


Figura 4.20: Detalle de la parte izquierda del mecanismo.

Una vez que se ha hallado la fórmula que define a α para este subconjunto, se puede analizar el otro lado del mecanismo que es el que depende de β. Este ángulo se puede descomponer en otros dos α' y β' que al restarlos dan como resultado el valor buscado. Como se acaba de ver, α' queda definido por l_i, en este caso, y por $-d_h$. Sin embargo, β' queda declarada como la apertura entre la vertical y el eslabón variable l_i', al colocar su extremo a una altura igual a la que se encuentre el extremo de l_d y a una distancia horizontal igual a la diferencia entre el desplazamiento *desp* menos el parámetro geométrico d_h.

$$\beta = \beta' - \alpha' \tag{4.48}$$

$$\alpha' = \arcsin \frac{-d_h}{l_i} \tag{4.49}$$

$$\beta' = \arcsin \frac{desp - d_h}{l_i'} \tag{4.50}$$

A priori, β' no queda totalmente definida, al depender de la variable l_i' que aun no conocemos su valor. Empezando por esto, para poder declarar definitivamente a β en función de sus dos ángulos, es necesario cuantificar de alguna manera el valor de l_i' en función de algún parámetro como α o *desp*. Para ello, y sabiendo que la cadera C es totalmente horizontal según los requerimientos, la altura de los dos extremos es la misma. De este modo, sabiendo cual es la altura en el extremo derecho, del cual tenemos todos los datos, se puede relacionar con la altura en el extremo izquierdo. Por lo tanto, se han de formular las dos alturas.

$$h_d = l_d \cos \alpha'' \tag{4.51}$$

$$h_i = l_i' \cos \beta' \tag{4.52}$$

$$l_d \cos \alpha'' = l_i' \cos \beta'$$

$$\cos\beta' = \frac{l_d}{l'_i}\cos\alpha'' \tag{4.53}$$

En este momento se tiene un sistema de ecuaciones en las cuales se tiene las incógnitas l'_i y β'. Para poder resolver este sistema solo es necesario dividir la ecuación 4.50 entre 4.53. Al hacer esto, se obtiene la siguiente ecuación.

$$\frac{\sin\beta'}{\cos\beta'} = \frac{\frac{desp - d_h}{l'_i}}{\frac{l_d\cos\alpha''}{l'_i}} \tag{4.54}$$

$$\tan\beta' = \frac{\frac{desp - d_h}{l'_i}}{\frac{l_d\cos\alpha''}{l'_i}} = \frac{desp - d_h}{l_d\cos\alpha''}$$

$$\beta' = \arctan(\frac{desp - d_h}{l_d\cos\alpha''}) \tag{4.55}$$

De este modo, β' queda totalmente definida en función de la variable *desp*, al estar α'' en función de esta misma variable. Por último, y simplemente por comodidad y estética, l_d y l_i tienen la misma longitud por motivos de diseño del robot, por lo que se puede sustituir una por la otra para dejar a β' relacionada con elementos de la parte izquierda del mecanismo. Con estas consideraciones, si también se desglosa α'', se puede dejar definida a β como lo siguiente.

$$\beta = \beta' - \alpha' = \arctan(\frac{desp - d_h}{l_d\cos\alpha''}) - \arcsin\frac{-d_h}{l_i} \tag{4.56}$$

$$\beta = \beta' - \alpha' = \arctan\left(\frac{desp - d_h}{l_d\cos(\arcsin\frac{d_h + desp}{l_d})}\right) - \arcsin\frac{-d_h}{l_i} \tag{4.57}$$

Una vez que se tienen perfectamente definidos los ángulos en función de parámetros conocidos, solo se necesita saber el acortamiento final que se produce en el eslabón l_i. Como se ha visto antes, l_i es partícipe en las ecuaciones 4.53 y 4.50, por lo que para que haya este valor sólo es necesario despejar de una de ellas. En particular, se despejará l_i a partid de la primera de estas dos ecuaciones.

$$\cos\beta' = \frac{l_d}{l'_i}\cos\alpha''$$

Se puede sustituir el valor l_d por l_i por tener las mismas características según lo ya visto.

$$\cos\beta' = \frac{l_i}{l'_i}\cos\alpha''$$

$$l'_i = l_i\frac{\cos\alpha''}{\cos\beta'} \tag{4.58}$$

Como se puede ver en la ecuación 4.58, la proporción que se acorta la pierna es igual a la relación $\frac{\cos\alpha''}{\cos\beta'}$. Como se ha dicho antes, la pierna ha de encogerse siempre, por lo que el la relación debe tomar un valor entre cero y uno. Para que esto se produzca, α'' ha de ser en

todo momento mayor que β'. Como se puede ver en 4.59, esta relación se puede ver mejor si se desglosan ambos ángulos.

$$\frac{l'_i}{l_i} = \frac{\cos\left(\arcsin\dfrac{d_h + desp}{l_i}\right)}{\cos\left(\arctan\left(\dfrac{desp - d_h}{l_d\cos(\arcsin\dfrac{d_h + desp}{l_i})}\right)\right)} \tag{4.59}$$

Si se atiende únicamente a los términos $desp + d_h$ y $desp - d_h$ se puede ver que, en la condición de $desp \geq 0$, el primer termino es siempre mayor. Pese a que el término $desp - d_h$ esta incluido en $\dfrac{desp - d_h}{l_d\cos(\arcsin\dfrac{d_h + desp}{l_i})}$ y hace que no sea lineal, sea cual sea el valor de $desp$, siempre que se ciña a la restricción anterior, siempre provocará que el termino correspondiente al ángulo del nominador sea mayor que el del denominador.

Por último y basándose en los visto en la sección 4.2.4 relativa al *movimiento de rodilla*, se puede usar la implementación anterior para poder encoger la pierna según las restricciones impuestas. Como se ha visto anteriormente en la fórmula 4.21, esta depende de una relación para el acortamiento de la forma $\dfrac{(L-R)-A}{(L-R)}$. En pocas palabras, esta relación siempre da un número entre uno y cero por relacionar la longitud de la pierna acortada con la longitud de la pierna estirada.

Pese a que los ejes de rotación de los eslabones del *movimiento de rodilla* no coincidan en el espacio, la fórmula relaciona siempre la longitud de la que se parte con a la que se desea llegar. Aunque l_i y L están definidas por ejes diferentes, se puede seguir empleando la ecuación 4.7. En este caso, para el acortamiento de la pierna izquierda ha de seguir todo lo visto en el apartado 4.2.4 *movimiento de rodilla* pudiendo sustituir en la fórmula 4.21 el término $\dfrac{(L-R)-A}{(L-R)}$ por $\dfrac{l'_i}{l_i}$, según 4.59.

$$\alpha_{rodilla} = \arccos\left(\frac{\cos(\arcsin\dfrac{d_h + desp}{l_i})}{\cos\left(\arctan(\dfrac{desp - d_h}{l_d\cos(\arcsin\dfrac{d_h + desp}{l_i})})\right)}\right)^{1,2} \tag{4.60}$$

[1] Se puede sustituir $\cos(\arctan(x))$ por $\dfrac{1}{\sqrt{1-x^2}}$

[2] Se puede sustituir $\cos(\arcsin(x))$ por $\sqrt{1-x^2}$, como se verá posteriormente

$$\alpha_{rodilla} = \arccos\left(\frac{\sqrt{1-(\frac{d_h+desp}{l_i})^2}}{\sqrt{1-\left(\frac{desp-d_h}{l_d\cdot\sqrt{1-(\frac{d_h+desp}{l_i})^2}}\right)}}\right)$$

Para finalizar y como aclaración, es importante recalcar que todos los subconjuntos se han de analizar por separado y, que aunque en este análisis esté involucrado el *movimiento de rodilla*, uno no depende de otro. Dicho de otro modo, aunque se haya que encoger la pierna usando para ello los mismos motores, los dos movimientos son independientes y dependen de un parámetro de entrada diferente para cada uno de ellos.

Versión de cadera estrecha

En esta versión de diseño del robot se ha conseguido alinear verticalmente los ejes q_2 y q_7 de cada pierna. Este cambio provoca que el paralepípedo a estudiar se convierta en un rectángulo cuando el robot está en reposo, como se puede ver la en figura 4.21. La distancia que separa a los tobillos es la misma que la distancia entre las uniones de la pierna con la cadera, siendo en ambos casos $81mm$. Al quedar completamente alineados verticalmente, la distancia entre los ejes superiores e inferiores de cada pierna pierna solo tienen componente vertical, correspondiente a un valor de $110.5mm$.

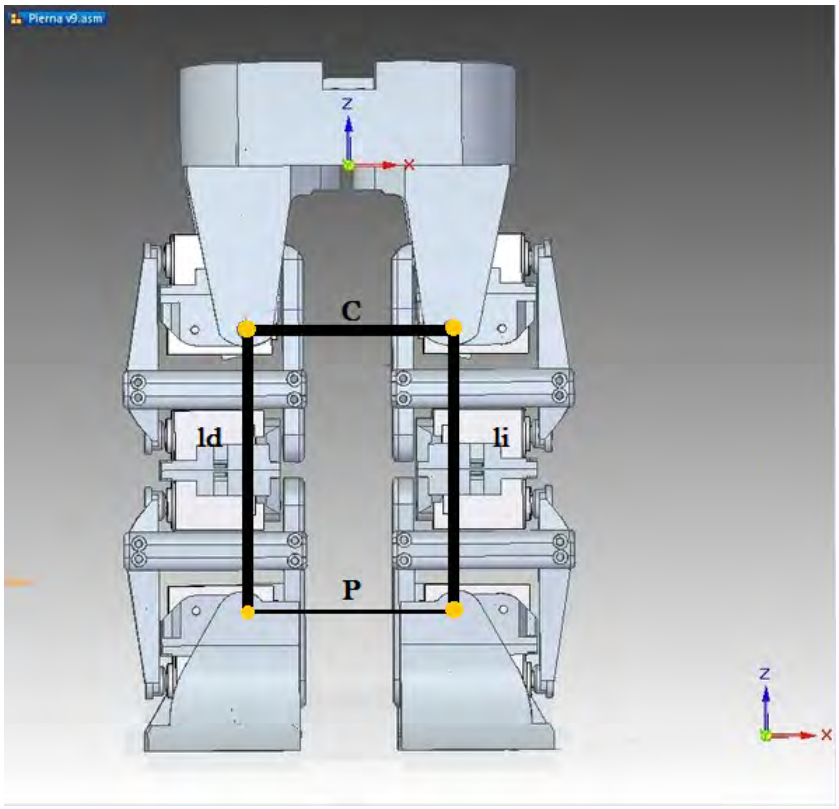


Figura 4.21: Presentación esquemática de la versión de cadera estrecha.

De igual modo que en la configuración anterior, para estudiar este subconjunto es necesario

analizar las dos piernas a la vez y teniendo en cuenta que ambas están unidas por la cadera, que también es necesario incluirla. En este caso, la cadena cinemática se simplifica notablemente, al no tener ángulos y segundas componentes en las distancias que definen los elementos. Para esta configuración, se puede definir los elementos que participan en él del siguiente modo.

Eslabones del paralepípedo

Eslabón cadera C.

Este eslabón está situado en la parte superior de la figura teniendo en los extremos a los ejes q_2 de cada pierna. Por motivos geométricos, al no existir ningún elemento en el plano de alzado que pueda variar esto, la distancia entre ambos ejes siempre será de 81 mm, quedando definidas las dimensiones del eslabón.

Eslabón distancia entre tobillos P

Esta distancia corresponde con el eslabón menor de cuerpo. Este eslabón se considera totalmente inmóvil, sin poder desplazar ni rotar su posición. A los extremos de él se encuentran ambos ejes q_7, que se separan una distancia de 81 mm, que coincide en este caso con la distancia de C. Pese a poder variar esta distancia al rotar los ejes q_7 o al acotar la pierna girando los ejes q_3, q_4, q_5 y q_6, se impone como condición del estudio que no se pueda variar de ningún modo. Esta condición emula que ambos pies no se separen del suelo y que el movimiento final esté destinado a la cadera.

Eslabón pierna derecha l_d

Este eslabón queda definido por la unión en línea recta de los ejes q_2 y q_7 de la pierna derecha, desde ahora q_{d2} y q_{d7}. La distancia de este eslabón es $110.5mm$ en vertical, sin existir una componente horizontal. Hay que señalar que l_d no es igual a L, al ser otra distancia medida sobre ejes diferentes. No obstante, por motivos de diseño, estos parámetros están relacionados por $l_d = L - 2R$. Esta distancia l_d si que puede variar al poder definir como condición de contorno, en el caso que se quiera, al poder articular la rodilla de forma similar a lo visto en el apartado correspondiente a *movimiento de rodilla*.

Eslabón pierna derecha l_i

De modo similar al anterior, este eslabón esta definido por los ejes q_2 y q_7 de la pierna izquierda, q_{i2} y q_{i7}. Por ser idéntico al derecho, cumple las mimas condiciones y características que l_d.

Para este análisis, del mismo modo que se ha hecho para la versión de *cadera ancha*, se va a intentar mover la cadera de dos formas diferentes. En el primer caso, la cadera solo podrá desplazarse horizontalmente, mientras que en la segunda podrá existir un giro de la cadera que pueda permitir el desplazamiento y giro del torso del robot. Además, como se podrá ver, esto último se podrá realizar de dos modos diferentes, pudiendo variar la altura de uno o los dos puntos extremos de la cadera.

Movimiento de cadera horizontal Una vez definido el paralelogramo que se usará en el análisis se puede empezar a estudiar una nueva forma de movimiento para la cadera. Esta cadena cinemática sigue siendo un mecanismo con un solo grado de libertad, por lo que existe muchas restricciones que caracterizan la interactuacción de los elementos entre sí.

Como particularidad, si se rota una barra lateral de este paralelogramo, el giro en las cuatro esquinas tendrá el mismo módulo y sentido diferente en la pareja superior que en la inferior. Esto además ocasiona que la barra superior, siempre que los pies no se muevan del suelo, queda siempre en posición horizontal sin importar en que posición del espacio ocupe. Por último, la otra barra lateral también girará el mismo ángulo que la barra que provoque el movimiento. Este efecto se produce por tener lados iguales dos a dos en los lados opuestos de la figura.

Basándose justo en esto, se puede definir un desplazamiento horizontal *desp* que produzca un giro relativo de cada uno de los elementos rotativos. Este estudio es muy sencillo, ya que

muchos de los parámetros que se han tenido que tener en cuenta en los estudios relativos a la versión de *cadera ancha* desaparecen.

Del mismo modo que lo que se ha visto hasta ahora para los estudios para el *movimiento de cadera*, el sentido positivo de giro será considerado el sentido horario. También, para una visualización más simple del conjunto, el plano en el que encuentran los elementos es el plano de alzado, visto desde atrás.

Como se ve en la figura 4.22, es muy sencillo analizar que el desplazamiento y el ángulo girado están relacionadas por la longitud de los eslabones laterales. De tal modo que se puede formular el triángulo generado con la fórmula 4.61

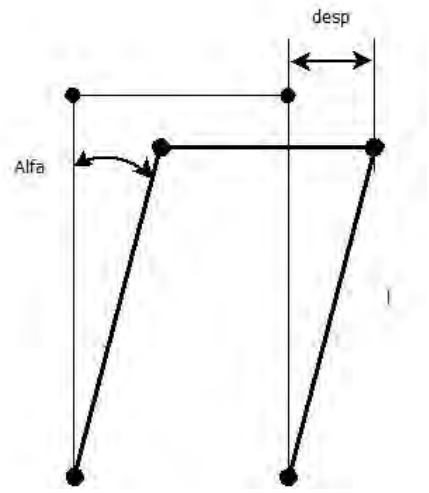


Figura 4.22: Descripción del movimiento de cadera horizontal.

$$\sin\alpha = \frac{desp}{l_d} \tag{4.61}$$

Por lo tanto, despejando α, se puede saber cuando se ha de girar cada ángulo dependiendo del desplazamiento.

$$\alpha = \arcsin\frac{desp}{l_d} \tag{4.62}$$

Si este mismo giro se produce en el otro eslabón lateral, la altura de ambas barras es idéntica. Esto es lo que produce que no exista un giro relativo en el eslabón superior. Para que esto se produzca, además, los ejes que unen la cadera con los eslabones laterales han de compensar el giro que se produce en los ejes de los tobillos para poder seguir manteniendo la orientación de los extremos. Por lo tanto, se puede expresar lo siguiente.

$$q_{d7} = \alpha$$

$$q_{i7} = \alpha$$

$$q_{d2} = -\alpha$$

$$q_{i2} = -\alpha$$

Relación entre ambas versiones imponiendo $d_h = 0$. Si se analiza este estudio y se relaciona con los dos relativos a la versión de *cadera ancha* se puede concluir que este estudio es un caso particular de cualquiera de los otros dos estudios previos. Si se analizan cualquiera de las fórmulas que relacionan los giros de cualquier estudio con los desplazamientos y d_h se sustituye por cero, se puede conseguir la fórmula de giro de este estudio.

En primer lugar, para comprobar esto, se sustituirá en las fórmulas 4.35 y 4.33 correspondiente a los valores de α y β del estudio hecho para el *giro de cadera con desplazamiento* para la versión de *cadera ancha*.

Para empezar β corresponde con

$$\beta = \arcsin \frac{2d_h \sin \alpha}{C}$$

$$\beta = \arcsin \frac{2(0) \sin \alpha}{C}$$

$$\beta = \arcsin(0)$$

$$\beta = 0$$

Por otro lado, α se despeja apoyándose en $\beta = 0$.

$$\alpha = \arcsin[\frac{d_h}{l'}(\cos \alpha - 1) + \frac{desp}{l'} - \frac{(\cos \beta - 1)C}{2l'}]$$

$$\alpha = \arcsin[\frac{0}{l'}(\cos \alpha - 1) + \frac{desp}{l'} - \frac{(\cos(0) - 1)C}{2l'}]$$

$$\alpha = \arcsin[0 + \frac{desp}{l'} - \frac{(1-1)C}{2l'}]$$

$$\alpha = \arcsin[0 + \frac{desp}{l'} - \frac{(0)C}{2l'}]$$

$$\alpha = \arcsin[0 + \frac{desp}{l'} - 0]$$

$$\alpha = \arcsin[\frac{desp}{l'}]$$

En este momento se puede afirmar que l', con la condición de $d_h = 0$, es igual a l_d. Para ello hay que basarse en la ecuación 4.24

$$l_d = \sqrt{l'^2 + d_h^2}$$

$$l_d = \sqrt{l'^2 + 0^2}$$

$$l_d = \sqrt{l'^2}$$

$$l_d = l'$$

Por lo tanto, α queda definida del mismo modo.

$$\alpha = \arcsin(\frac{desp}{l_d})$$

Por último, θ, según la fórmula 4.36, relativa a los ejes superiores queda como

$$\theta = -\alpha + \beta$$

$$\theta = -\alpha$$

Como ya se ha dicho, se puede comprobar que la premisa de que este estudio es una caso particular de los anteriores es cierta.

Además, se puede hacer lo mismo con el estudio de *movimiento horizontal de cadera* de la otra versión. Para ello, se ha de partir de las ecuaciones 4.47, 4.57 y 4.58.

$$\alpha = \alpha'' - \alpha' = \arcsin \frac{d_h + desp}{l_d} - \arcsin \frac{d_h}{l_d}$$

$$\alpha = \arcsin\frac{0+desp}{l_d} - \arcsin\frac{0}{l_d}$$

$$\alpha = \arcsin\frac{desp}{l_d} - \arcsin(0)$$

$$\alpha = \arcsin\frac{desp}{l_d}$$

En el caso de β

$$\beta = \beta' - \alpha' = \arctan(\frac{desp - d_h}{l_d\cos(\arcsin\frac{d_h+desp}{l_d})}) - \arcsin\frac{-d_h}{l_i}$$

$$\beta = \arctan(\frac{desp - 0}{l_d\cos(\arcsin\frac{0+desp}{l_d})}) - \arcsin\frac{0}{l_i}$$

$$\beta = \arctan(\frac{desp}{l_d\cos(\arcsin\frac{desp}{l_d})})^3$$

$$\tan\beta = \frac{desp}{l_d}\cdot\frac{1}{\sqrt{1-(\frac{desp}{l_d})^2}}$$

$$\tan\beta = \sin(\alpha)\cdot\frac{1}{\sqrt{1-(\sin\alpha)^2}}$$

$$\tan\beta = \sin\alpha\cdot\frac{1}{\cos\alpha}$$

$$\tan\beta = \tan\alpha$$

Por lo que

$$\alpha = \beta$$

Por último, la relación de acortamiento de la pierna ha de ser igual a uno, para que esta no varíe su longitud. Para demostrar esto hay que apoyarse en la ecuación 4.58.

$$l_i' = l_i\frac{\cos\alpha''}{\cos\beta'}$$

$$l_i' = l_i\frac{\cos(\alpha+\alpha')}{\cos(\beta+\alpha')}$$

[3]Para resolver el siguiente paso es necesario demostrar que $\cos(\arcsin(x)) = \sqrt{1-x^2}$. Para ello hay que basarse en un triángulo rectángulo de hipotenusa igual a uno y el cateto opuesto igual a x.

$$\sin\theta = x/1$$

$$\theta = \arcsin(x)$$

Por otro lado

$$cos\theta = \sqrt{1-(\sin(x))^2}$$

Uniendo ambas partes

$$cos\arcsin(x) = \sqrt{1-(x)^2}$$

$$l'_i = l_i \frac{\cos(\alpha + \arcsin\frac{d_h}{l_d})}{\cos(\beta + \arcsin\frac{d_h}{l_d})}$$

$$l'_i = l_i \frac{\cos(\alpha + \arcsin\frac{0}{l_d})}{\cos(\beta + \arcsin\frac{0}{l_d})}$$

$$l'_i = l_i \frac{\cos(\alpha + 0)}{\cos(\beta + 0)}$$

$$l'_i = l_i \frac{\cos(\alpha)}{\cos(\alpha)}$$

$$\frac{l'_i}{l_i} = 1$$

De esta forma queda finalmente demostrado que este estudio es un caso particular de cualquiera de los anteriores relativos al movimiento de la cadera en los que se imponga la restricción $d_h = 0$. Siguiendo lo mismo, se puede demostrar que se pueden usar las mismas ecuaciones para $d_h < 0$, salvo para la relación de acortamiento 4.58 que sería necesario volver a calcularla basándose en mover la pierna derecha en lugar de la izquierda, volviendo a obtener 4.58. Pero como no es una versión que este construida, diseñada o planteada para el robot se puede obviar en este momento.

Introducción al balanceo

Como se ha visto, el mecanismo paralepípedo que se analiza en este estudio presenta la característica de que la barra superior es siempre paralela al suelo, por este motivo no se puede producir ningún giro local en la barra ni de cualquier otro elemento que esté sujeto a esta. En ocasiones puede interesar producir este giro relativo para desplazar el peso del robot, sobre todo en los casos de que el centro de gravedad esté por encima de la cadera.

Ya que este mecanismo no permite este giro, es necesario incluir un nuevo grado de libertad para poder provocar este efecto. El robot al no tener un eje de giro alojado sobre el abdomen o la cadera de esté que permita rotar el torso en el plano frontal, se puede recurrir al estiramiento y acortamiento de una o las dos piernas para poder rotar la cadera, como se ve en la figura 4.23.

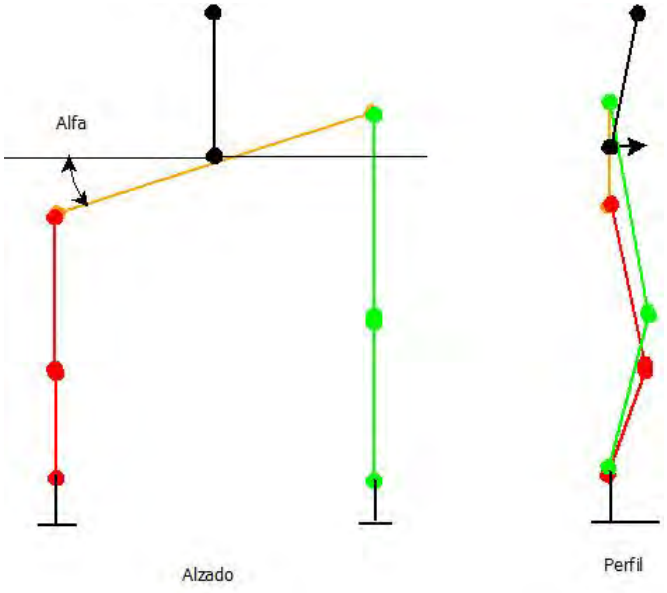


Figura 4.23: Emulación de la articulación de la cadera al flexionar las piernas.

Una pierna estirada En primer lugar, para conseguir imitar el abdomen de el robot que permita el giro del torso se comenzará encogiendo una sola pierna. Al encoger una pierna y dejar la otra completamente estirada, la altura total de estas es diferente. Si se considera que estos puntos son los extremos de cadera C, al existir una diferencia de alturas, se produce una rotación en dicho eslabón.

Para conseguir esto, es necesario acortar la longitud total de una de las dos piernas entre los ejes q_2 y q_7. A diferencia de lo visto hasta ahora, el parámetro maestro que controle este movimiento será un ángulo α en lugar de una distancia. Este parámetro indicará en que ángulo se ha de colocar la cadera C respecto al suelo. Además, si uno es consciente de ello, la cadera al rotar pierde parte de su longitud horizontal. Al estar impuesto como condición de contorno que los pies no pueden separarse ni alejarse del suelo, la distancia entre tobillos es mayor que la distancia horizontal de la cadera. Para que todo el mecanismo siga unido sin violar ninguna condición de contorno, es necesario rotar el eje de un tobillo para que el extremo superior de la pierna siga estando en la misma posición del plano que el extremo de la cadera correspondiente. Arbitrariamente, pudiendo elegir si se desea la otra opción o rotar ambos tobillos para que el centro de la cadera C_o siga proyectándose en el centro de P, se ha decidido variar el ángulo del tobillo de la pierna que no se acorta.

Si se quiere producir este acortamiento es necesario doblar la pierna según lo visto anteriormente en *movimiento de rodilla*. Para indicar este acotamiento es necesario que exista una relación entre la pierna con la posición deseada y la pierna cuando se la considere estirada, además de quedar en función de α.

Como la pierna que se encoge está en función del signo del ángulo α, solo se analizará uno de estos dos casos, en concreto cuando $\alpha \geq 0$. En esta condición, la pierna que se encoge es la izquierda y la que ha de girar el tobillo es la derecha. Para el caso de que $\alpha \leq 0$, los roles de las piernas son los contrarios.

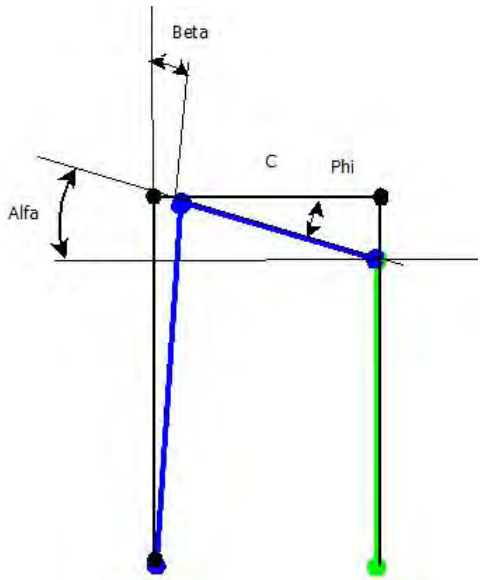


Figura 4.24: Disposición de los elementos en el estudio de una pierna estirada.

En primer lugar, se define un giro α que se desea para la cadera. Tras esto, interesa saber como cambian las longitudes horizontales y verticales para el eslabón de la cadera C.

$$\Delta x_c = C\cos\alpha \tag{4.63}$$

$$\Delta y_c = C\sin\alpha \tag{4.64}$$

Por lo tanto, la diferencia dif que se ha producido en la longitud horizontal de la cadera es

$$dif = C - C\cos\alpha = C(1-\cos\alpha) \tag{4.65}$$

A partir de este parámetro y de la longitud total del eslabón derecho, se puede obtener que ángulo gira el tobillo derecho β.

$$\sin\beta = \frac{dif}{l_d}$$

$$\beta = \arcsin\frac{C(1-\cos\alpha)}{l_d} \tag{4.66}$$

Con las relaciones hasta ahora obtenidas son suficientes para obtener cuando se acota la pierna izquierda. Para ello es necesario centrarse únicamente en la componente vertical de los elementos de ambas piernas y la cadera. Para que estos elementos estén unidos entre sí, la suma de la altura de los elementos por cada lado ha de ser igual. Partiendo de esto, la longitud de la pierna izquierda encogida l_i' se puede definir como lo siguiente.

$$l_i' = l_d\cos\beta - \Delta y_c = l_d\cos\beta - C\sin\alpha \tag{4.67}$$

Por último, es necesario obtener el giro que se producen en los ejes de la cadera. Para ello, si uno se fija en la figura 4.24, se puede ver que q_i2 ha de girar un ángulo α para que este pueda orientar la cadera de forma correcta. Finalmente solo falta por definir el giro que se produce en el extremo opuesto de la cadera. Para facilitar la visión de esto, es necesario centrarse en la unión entre la pierna derecha y el extremos del mismo lado de la cadera, tal y como se representa en la figura 4.25.

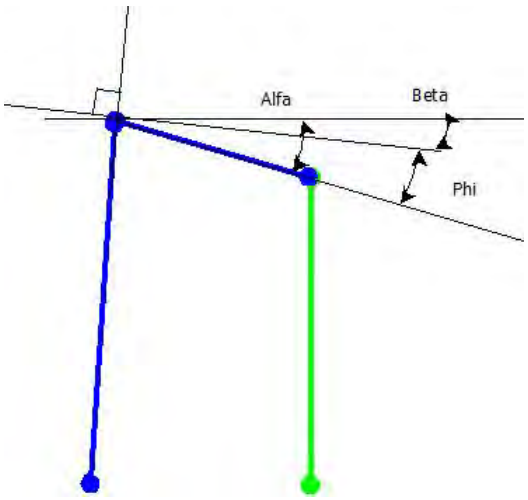


Figura 4.25: Detalle de los ángulos presentes.

Si se relacionan todos los ángulos presentes en la parte superior de la cadera y que todos partan de q_d2, se puede relacionar el giro correspondiente a este eje φ con α y β.

$$\varphi = \alpha - \beta \tag{4.68}$$

Con esta última ecuación quedan todos los elementos relacionados por los parámetros geométricos presentes y por α. Ahora solo es necesario decretar que ángulo corresponde con cada giro. Para refrescar lo anterior y tener agrupado se muestras las siguientes relaciones.

$$q_i7 = 0$$

$$q_d7 = \beta$$

$$q_i2 = \alpha$$

$$q_d2 = \varphi$$

Y por último, atendiendo a la ecuación 4.21 correspondiente al *movimiento de rodilla*, se puede obtener los ángulos α que han de girar los ejes relacionados con el acortamiento.

$$\alpha_{rodilla} = \arccos(\frac{l'_i}{l_i}) = \arccos(\frac{l_d \cos\beta - C \sin\alpha}{l_d})$$

Ambas piernas encogidas Otra forma de rotar la cadera con esta versión que se va a ver en este proyecto consiste en encoger ambas piernas. Para que se produzca de una forma controlada ambas piernas han de sincronizarse para que una aumente su longitud partiendo de una altura que se considerará el origen mientras que la otra se acorta. Esta altura de referencia depende del parámetro maestro α_{max} por lo que nunca es el mismo para valores diferentes. A diferencia de el resto de estudios, es necesario recurrir a un segundo parámetro que indique cual es la altura de referencia. Si se compara con el estudio anterior, existe un grado más de libertad al poder encoger dos piernas en lugar de una, por lo que hace pensar que es necesario esta nueva restricción.

Como se ha visto antes, la longitud horizontal de la cadera varía en función del giro α que se produzca, por lo que es necesario girar los tobillos para que ningún elemento quede separado de otro. En este caso, para simplificar un poco el cálculo y desplazar lateralmente el centro de gravedad de forma favorable, el tobillo de la pierna que esté más encogida no podrá girar.

La forma de relacionar los dos parámetros maestros α y α_{max} que definen al movimiento corresponde al que α no puede tener un módulo mayor que el de α_{max}.

$$-\alpha_{max} \leq \alpha \leq \alpha_{max} \tag{4.69}$$

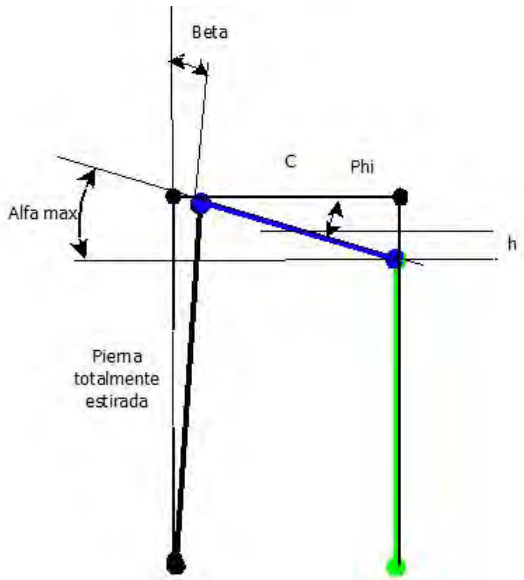


Figura 4.26: Disposición de los elementos en el estudio de ambas piernas encogidas.

Para afinar la altura de referencia es necesario referenciarla sobre la altura máxima que puede alcanzar la pierna estirada en función de α_{max}. En primer lugar, basta con calcular el incremento de alturas que se produce desde el centro de la cadera hasta un extremo del mismo.

$$h_{\Delta max} = \frac{C}{2} \sin \alpha_{max} \tag{4.70}$$

Esta altura queda referenciada al punto más alto de la pierna estirada, este punto está a una altura total correspondiente a la siguiente fórmula.

$$h_{max} = l_d \cos \beta_{max} \tag{4.71}$$

Donde β_{max} tiene un valor de

$$\beta_{max} = \arcsin(\frac{C(1 - \cos \alpha_{max})}{l_d}) \tag{4.72}$$

Sabiendo el punto más alto, ya queda totalmente definido la altura de referencia h_0. Partiendo de ella y sumando y restando el incremento máximo de altura se pueden obtener las alturas máximas y mínimas.

$$h_{max} = l_d \cos \beta_{max} = h_0 + h_{\Delta max} \tag{4.73}$$

$$h_{min} = h_0 - h_{\Delta max} = h_{max} - 2h_{\Delta max} \tag{4.74}$$

Si se despeja de 4.73 se puede obtener la fórmula de la altura de referencia.

$$h_0 = h_{max} - h_{\Delta max} = l_d \cos \beta_{max} - \frac{C}{2} \sin \alpha_{max} \tag{4.75}$$

El centro de la cadera C_0 ha de estar en todo momento a esta altura h_0, sin importar la posición en el eje X o la orientación α de la cadera C, siempre que cumpliendo el resto de restricciones. Una vez que se ha definido el giro máximo permitido al imponer el valor máximo que se quiere que tome, se puede analizar un modelo genérico para el robot. Para ello, y apoyándose en 4.27 en la que se muestra el conjunto con una restricción α_{max} y con un giro genérico α, se puede analizar todas las relaciones que se producen. Para comenzar con el estudio de la cadena formada, el mejor punto de partida es la diferencia vertical de los extremos de la cadera con la imposición de $y_{C_0} = h_o$.

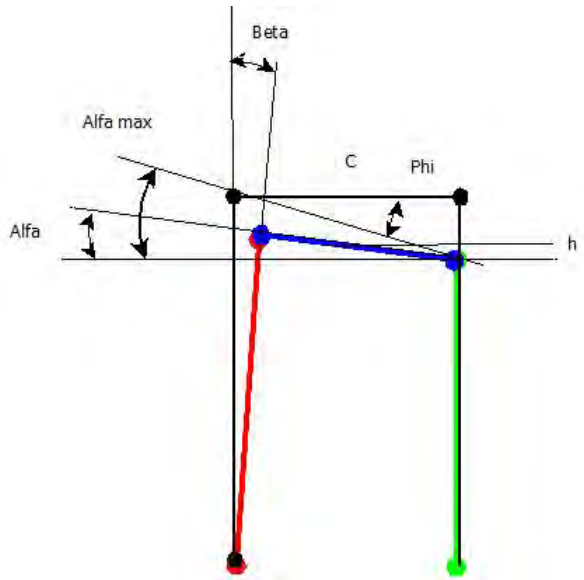


Figura 4.27: Movimiento de la estructura ante un giro α intermedio.

$$\Delta y_C = C \sin \alpha \tag{4.76}$$

Al rotar sobre el punto medio al poner la imposición $y_{C_0} = h_o$, se puede dividir este incremento a partes iguales. Al sumar y restar estos semiincrementos a la distancia vertical de cada pierna, se puede ver cual es la longitud de estas.

$$h'_{l_d} = h_o + \frac{1}{2}\Delta y_C = l_d \cos \beta_{max} - \frac{C}{2} \sin \alpha_{max} + \frac{1}{2} C \sin \alpha \tag{4.77}$$

$$h'_{l_i} = h_o - \frac{1}{2}\Delta y_C = l_d \cos \beta_{max} - \frac{C}{2} \sin \alpha_{max} - \frac{1}{2} C \sin \alpha \tag{4.78}$$

Si se recuerda la imposición de que ante un $\alpha \geq 0$ la pierna que más se encoge es la izquierda y, que por lo tanto, su tobillo no puede rotar. Esto quiere decir que el eslabón no presenta ningún tipo de distancia horizontal, siento $h'_{l_i} = h_{l_i}$

$$h_{l_i} = l_d \cos \beta_{max} - \frac{C}{2}[\sin \alpha_{max} - \sin \alpha] \tag{4.79}$$

Antes de centrarse en la longitud real de la parte derecha, es necesario calcular el giro del tobillo derecho β. Para saber cual es su definición, es necesario apoyarse en la altura del eslabón derecho h'_{l_d} y el acortamiento que se produce en horizontal en la cadera Δx_c.

$$\beta = \arctan \frac{\Delta x_c}{h'_{l_d}} = \arctan\left(\frac{C(1 - \cos \alpha)}{l_d \cos \beta_{max} - \frac{C}{2}[\sin \alpha_{max} - \sin \alpha]}\right) \tag{4.80}$$

Tras saber el valor de β en función de α para el caso concreto de α_{max} se puede hallar la longitud real de la pierna derecha. Para ello basta con relacionar la distancia vertical con la longitud real a través de β.

$$\cos \beta = \frac{h'_{l_d}}{h_{l_d}}$$

$$h_{l_d} = \frac{h'_{l_d}}{\cos\beta} = \frac{l_d\cos\beta_{max} - \frac{C}{2}[\sin\alpha_{max} + \sin\alpha]}{\cos(\arctan(\frac{C(1-\cos\alpha)}{l_d\cos\beta_{max} - \frac{C}{2}[\sin\alpha_{max} - \sin\alpha]}))}^{4} \tag{4.81}$$

Por último, y partiendo de 4.68 al partir de las mismas premisas, se puede obtener el último ángulo que define al sistema que está situado en la unión de la pierna derecha con la cadera.

$$\varphi = \alpha - \beta$$

Para poder utilizar las relaciones vistas en *movimiento de rodilla* y poder aplicarlas a esto es necesario tener en cuenta algunas cosas. En primer lugar, la ecuación relaciona mediante una fracción dos longitudes. Hasta ahora, siempre se ha visto que en el denominador aparece un número fijo que en todo momento es mayor que el nominador, esto produce una relación que siempre da valores entre cero y uno, si se tienen en cuenta otras restricciones. Además, por ser una función basada en $\arccos(x)$, el argumento de la función no puede superar la unidad, ya que la función no da una solución real.

Hay que tener en cuenta que la longitud de las piernas en h_0 no es la máxima que pueden alcanzar, por lo que no se puede tomar esta referencia para ponerla en el denominador porque habrá casos en los que esta relación supere la unidad. Por lo tanto, la forma correcta de definir esta fracción es poner en el denominador la longitud real de la pierna cuando está se considere estirada, por lo que se puede definir, basándose en 4.21, como

$$\alpha_{rodilla_d} = \arccos(\frac{h_d(\alpha)}{h_{max}}) = \arccos(\frac{h'_{l_d}}{h_0 + h_{\Delta max}}) \tag{4.82}$$

Siendo idéntica para la pierna izquierda salvo por tener en el nominador el término relativo a dicha pierna.

4.2.6. Subconjunto del *movimiento de rotación de la cintura*

Con este estudio se pretende poder girar de forma relativa la cadera con el torso. Para que el esto de elementos que tiene ejes paralelos a este, como los grados de libertad q_1 de las dos piernas y el cuello, no se vean afectados han de contraponerse a este giro para que mantengan sus orientaciones.

Al no tener el origen de todos los ejes en un mismo plano, es difícil una representación de cada uno en un plano en el que se vea realmente el giro de todos ellos. No obstante, se puede ver una representación gráfica del movimiento de estos motores en la figura 4.28 Además al no haber ningún otro tipo de elemento a parte de los citados ejes, este subconjunto queda definido por giros, sin importar las distancias o coordenadas de otros elementos.

[4] Se puede sustituir $\cos(\arctan(x))$ por $\frac{1}{\sqrt{1-x^2}}$

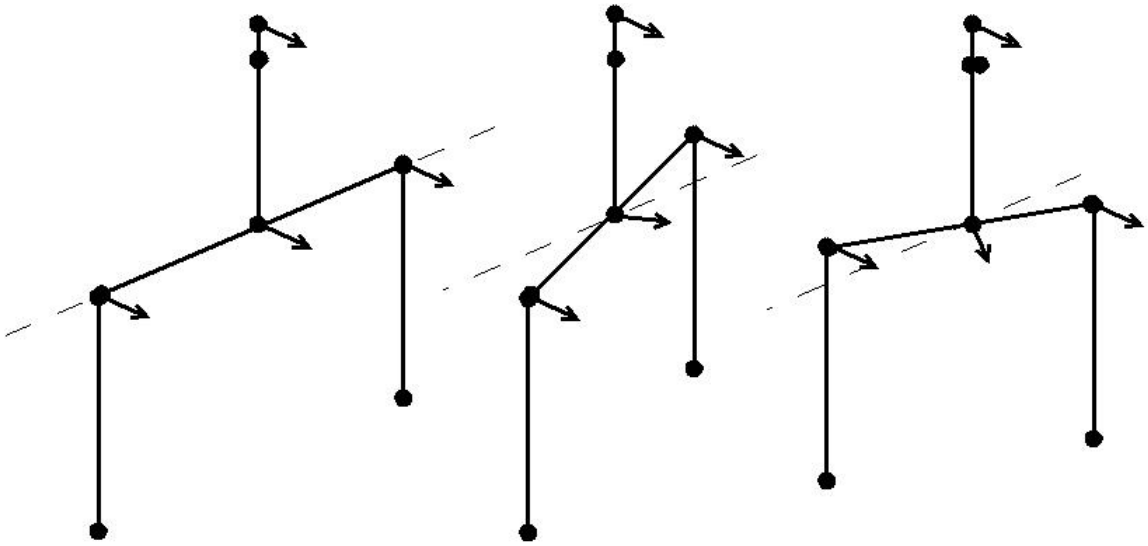


Figura 4.28: Movimiento de cintura.

El parámetro maestro de este estudio es la rotación que se produce entre el torso y la cadera α en el eje de la cintura $q_{cintura}$. Como se acaba de ver, el resto de ejes se contraponen a este giro para no perder la orientación. No todos los servomotores tienen su piñón apuntando en la misma dirección, por lo que es necesario tener en cuenta que el cuello apunta hacia arriba mientras que el resto lo hacen hacia abajo. Respetando esto último, y al saber que todos los motores que participan son *TowerPro MG996R* 2.1, salvo el del cuello que es *TowerPro MG9S* 2.3, se puede ver el giro de cada uno de ellos en las ecuaciones siguientes.

$$q_{cintura} = \alpha \tag{4.83}$$

$$q_{d1} = q_{i1} = -\alpha \tag{4.84}$$

$$q_{cuello} = \alpha \tag{4.85}$$

4.2.7. Subconjunto del *movimiento de brazos*

El último subconjunto que se analizará será el de *movimiento de brazos*. Este movimiento de brazos permite compensar hacia adelante y atrás el centro de gravedad del robot. La inestabilidad del cuerpo en este eje se produce porque las piernas desplazan en su movimiento al *CDG*. Para compensar esto, los humanos tendemos a avanzar el brazo opuesto a la pierna adelantada y retrasar el otro, equilibrándonos en cada paso.

Imitando a esto, el *movimiento de brazos* permite balancear los brazos del robot de forma similar a la nuestra. El robot no presenta una articulación que emule a un codo humano, por lo que este movimiento no imita a la perfección al antropomórfico. Por la falta de este eslabón, el brazo queda configurado como una barra rígida, la cual rota sobre su extremo superior.

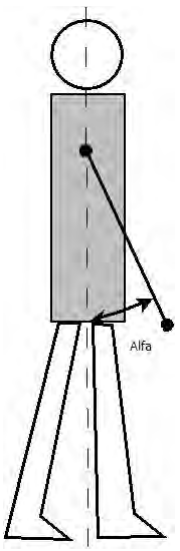


Figura 4.29: Movimiento de brazos.

Antes de iniciar el análisis, se ha de señalar que dependiendo de la herramienta que lleve el robot en el extremo del brazo esta puede interferir o no en las pierna al colisionar con ellas. Para evitar esto, es necesario variar levemente el ángulo del brazo medido sobre la vertical. Este ángulo provoca que el extremo del brazo se separe cierta distancia, evitando la colisión de la herramienta con la pierna en la trayectoria que pueda tener al basarse en este estudio. De forma arbitraria se puede cambiar esta separación al definir una posición en el segundo grado de libertad q_2 de cualquier brazo.

Como se ha mencionado, el brazo se queda configurado como una barra rígida que rota sobre uno de los extremos. Esto emula de nuevo a un mecanismo con forma de péndulo, pudiéndolo analizar de una forma muy sencilla. El brazo rota sobre el eje de la primera articulación q_1. Este giro provoca el desplazamiento de todo el brazo, cambiando la orientación en el plano lateral del extremo.

Desgraciadamente, el robot no presenta una única herramienta que puede portar en el extremo, no pudiendo definir una distancia total del brazo, del mismo modo que tampoco se puede calcular la masa total para hacer un estudio estático o dinámico. Por esta falta de definición no se puede concretar de una forma específica ninguno de estos parámetros, por lo que tampoco se puede formular una cinemática directa ni inversa para el extremo del robot.

Para provocar el movimiento, sin poder calcular específicamente el punto donde se colocará el extremo del robot por este no estar definido, el movimiento del brazo solo quedará calculado en función de q_1 de forma exclusiva. Quedando θ en función de sí misma.

En ocasiones puede interesar definir un parámetro que haga que el giro no sea simétrico sobre la vertical, pudiendo adelantar más el brazo en dirección al pecho. Para poder producir esto, e imitar algo más el movimiento humano, se puede sumar otro ángulo a θ. Por lo tanto, este movimiento queda definido simplemente por 4.86.

$$q_1 = \theta + \delta \tag{4.86}$$

Del mismo modo que las piernas, este resultado se puede aplicar para el otro brazo, cambiando el signo para usarlo de un modo correcto.

También es importante tener en cuenta que tipo de motores se emplean para dicha articulación. Como se dijo al inicio, los brazos necesitan tener menores prestaciones que las piernas, por los que se pueden usar servomotores con una calidad inferior. Tal y como se menciona en las tablas 2.1 y 2.2, estos dos modelos de servos presentan diferentes sentidos de avance de giro.

Sabiendo que motor se usa, se puede prever en que sentido avanza el motor, siendo suficiente imponer un signo negativo o positivo para adecuar el giro al criterio impuesto.

4.3. Trayectorias de movimiento para los estudios de superposición

Apoyándose en los estudios de superposición, se puede mover al robot a lo lardo de una trayectoria en función de unos parámetros. Si se concatenan una serie de movimientos, se puede conseguir que el robot se mueva con fluidez, imitando un movimiento de caminata humana. Estos movimientos es necesario que sigan un progresión adecuada que no generen saltos o discontinuidades entre un movimiento y el siguiente, además de ser periódicos para poder usarlos de modo continuo fijándose únicamente en un periodo.

Como una caminata es siempre cíclica, se puede calcular el movimiento correspondiente a un solo periodo y repetirlo en bucle. Para esto es necesario que la trayectoria a seguir empiece y termine en un punto muy cercano al del inicio y que siga la progresión para que no se produzca una discontinuidad.

Para este proyecto se va a intentar buscar trayectorias para cada uno de estos elementos que sean lo más suavizadas posibles, pero evitar que los movimientos no sean fluidos y se produzcan picos en el torque de algún motor, que a su vez genere una sobrecorriente que pueda llegar a dañarlo.

En este proyecto se intentarán generar diferentes trayectorias que permitan al robot andar en línea recta, desplazarse lateralmente y girar sobre sí mismo. Con estos tres movimientos el robot puede desplazarse perfectamente sobre un espacio plano.

4.3.1. Trayectoria sinusoidal

Una de las trayectorias más suavizadas son las basadas en la función $\sin(\omega t + \varphi)$. Estas funciones no presentan discontinuidades a lo largo del tiempo, presentan una derivada y una derivada segunda también muy suavizadas al estar formadas por el mismo tipo de función. Además, esta función puede devolver valores tanto positivos como negativos en función del tiempo, y siempre en un rango que el módulo no supere en ningún caso la unidad.

Es muy sencillo parametrizar una trayectoria basada en este tipo de función, al poder multiplicar a estas por una amplitud A que corresponda con el valor extremo a alcanzar. También es igual de sencillo poder parametrizar esta función respecto al tiempo, pudiendo seleccionar fácilmente el periodo total en el que se quiere realizar el movimiento. Por último, se puede definir el desfase de esta función respecto a otra en la misma base de tiempo y el mismo periodo variando únicamente un solo parámetro del argumento. Como se ha planteado antes, al ser una función que se repite en el tiempo, solo se va a analizar entre 0 y 2π.

Modificaciones

En ocasiones se puede querer que el cinema pase más tiempo alojado en los extremos que en cualquier otro punto de la trayectoria. En el caso de usar una función sinusoidal, no existen dos instantes de tiempo contiguos en los que se repita un mismo valor, aunque en la región de los extremos los valores sean muy cercanos.

Interesa que, al igual que el resto de la función, los intervalos en los que la posición es la misma sigan provocando que la función sea simétrica. Esta simetría interesa que esté sobre el eje X y que el centro de los valores iguales a la unidad coincidan con el máximo y el mínimo de la función sinusoidal.

La forma más sencilla de hacer esta modificación es elegir el valor menor entre el módulo de la función o un valor umbral *val* entre cero y uno. Esto, de forma matemática se puede expresar

como

$$y = \text{mín}\left(|\sin(\omega t + \varphi)|, val\right) \cdot \frac{\sin(\omega t + \varphi)}{|\sin(\omega t + \varphi)|}$$

Si analizamos la función se puede ver que $\frac{\sin(\omega t + \varphi)}{|\sin(\omega t + \varphi)|}$ devuelve el signo correspondiente al seno de origen. También, el rango de valores pasa de [-1,1] a [-val,val]. Esto puede ser un problema grave, ya que si se usa esta expresión y se le obliga a ir a un cinema a la posición extrema A, este solo llegará a $A \cdot val < A$. Para poder utilizar este tipo de modificación, hay que devolver a la función al dominio [-1,1]. Para ello, es necesario multiplicar a la función por un escalar para que transforme de nuevo la función.

$$y = \frac{\text{mín}\left(|\sin(\omega t + \varphi)|, val\right)}{val} \cdot \frac{\sin(\omega t + \varphi)}{|\sin(\omega t + \varphi)|} \tag{4.87}$$

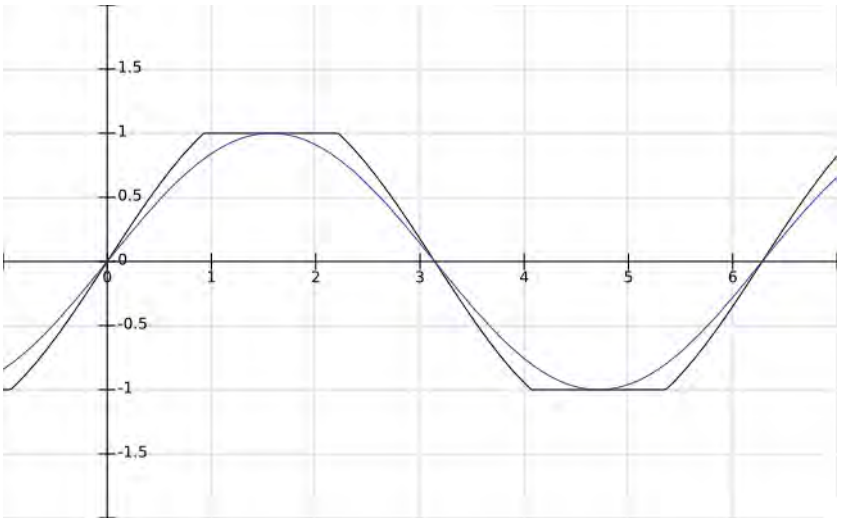


Figura 4.30: Función sinusoidal modificada (umbral 0.8).

Esta función sí está definida para el rango de valores deseado, presenta simetría y un suavizado en los extremos que puede ser parametrizable dependiendo del tiempo que se desee.

Por lo tanto, se puede definir una trayectoria basada en esta función con una amplitud máxima de A y un valor umbral de corte val en cualquier tiempo ωt_n entre 0 y 2π de la siguiente forma.

$$desp_n = A\frac{\text{mín}\left(|\sin(\omega t_n + \varphi)|, val\right)}{val} \cdot \frac{\sin(\omega t_n + \varphi)}{|\sin(\omega t_n + \varphi)|} \tag{4.88}$$

4.3.2. Trayectoria de pulso

En otras ocasiones puede interesar que todo el movimiento se concentre en un pequeño instante del tiempo y que luego regrese a la posición de reposo. Esta función puede asemejarse mucho a la *función δ de Dirac*. A diferencia de esta función, es necesario acotar al pulso a un valor máximo de uno en el momento que se desee y a cero el resto del tiempo.

$$y = \delta(t_n)$$

En la práctica, un impulso de un ancho de pulso tan pequeño no ocasionaría ningún efecto en el control de los motores, si se considera que existe un filtro a la entrada de los mismos y que

incluyen elementos con inercia que se resisten a un movimiento puntual. Por eso, es necesario utilizar un ancho de pulso mayor.

Como matematización de este ancho de pulso, podría mostrar esta señal como dos *funciones de Heaviside* en dos instantes diferentes en el tiempo y que se contrarresten tras el segundo instante.

$$y = 1 \cdot H(t_1) - 1 \cdot H(t_2) \qquad con \quad 0 < t_1 < t_2 < T$$

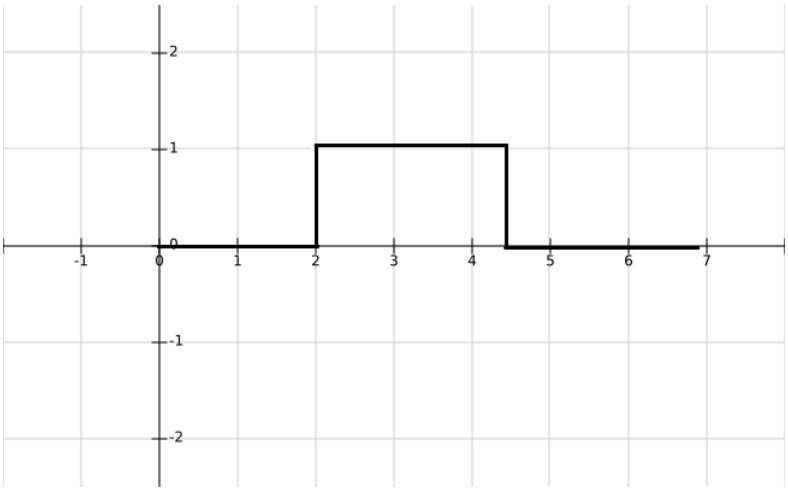


Figura 4.31: Función Heaviside Consecutivas que se anulan.

Las derivadas en los puntos t_1 y t_2 son infinitas para adaptarse al cambio. Esto en la práctica es imposible, generando un estado transitorio hasta alcanzar el valor objetivo. Al ser elementos mecánicos con un velocidad considerable pero no infinita, puede interesar más el definir punto a punto la trayectoria a seguir durante el cambio, en lugar de crear un transitorio que sólo puede controlar el lazo de control del motor.

Por este motivo, se puede considerar un suavizado en los instantes t_1 y t_2 de cada periodo e, incluso, generar una pendiente más suavizada para que los motores se adecuen en cada instante a la gráfica de un modo realista. Para ello, se puede usar este tipo de función.

$$y = (\sin(\omega t_n + \varphi))^n$$

Al multiplicar números menores a la unidad, se obtiene un número aun menor. Cuantas más veces se haga, menor será el número, por lo tanto, cuanto mayor sea n, mayor será la pendiente resultante y más tiempo pasará la función en un valor cercano al cero. Al tener siempre el seno el mismo argumento, existirán dos instantes de tiempo con valores de módulo igual a la unidad, al multiplicar uno n veces. En el resto de los casos, cuanto menor sea el módulo de cada seno, se podrá considerar que la función permanece en cero.

Uno de los problemas de esta función es que se generan dos picos en un mismo periodo y, el otro, que dependiendo de si n es par o impar se generan o dos picos positivos o uno de cada signo.

Para poder eliminar estos problemas, puede interesar obtener siempre dos picos con diferentes signo y eliminar el negativo. En el caso de que n sea par, los dos picos se producirán en el eje positivo, pudiendo no saber cual eliminar. Por lo tanto, es necesario homogeneizar la función para cualquier valor de $n > 0$.

$$y = (\sin(\omega t_n + \varphi)) \cdot |\, (\sin(\omega t_n + \varphi))^{n-1} \,|$$

En este caso, se consigue obtener siempre los picos con signos diferentes. El primer término se encarga de mantener el signo, mientras que del otro sólo interesa el valor de las multiplicaciones. En este momento, es sencillo eliminar el pico negativo, usando la siguiente función.

$$y = \text{máx}\left[\sin(\omega t_n + \varphi) \cdot |\left(\sin(\omega t_n + \varphi)\right)^{n-1}|,\ 0\right] \tag{4.89}$$

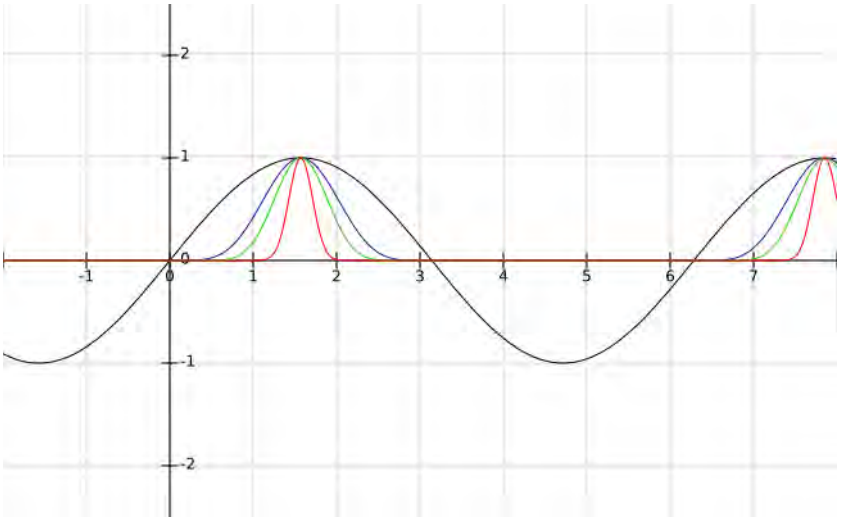


Figura 4.32: Función sinusoidal elevada. Exponentes 5,10 y 100.

El segundo subperiodo, por ser negativo, se elimina por la posición de reposo con valor nulo. De este modo, se consigue generar un impulso suavizado en en cada periodo. Esta función 4.89 se puede multiplicar por una amplitud máxima A y poder generar una trayectoria con la misma forma pero con un valor objetivo diferente.

$$desp_n = A \cdot \text{máx}\left[\sin(\omega t_n + \varphi) \cdot |\left(\sin(\omega t_n + \varphi)\right)^{n-1}|,\ 0\right]$$

4.3.3. Trayectoria triangular

Pueden existir ocasiones en los que la trayectoria que más interese sea una que llegue a dos puntos simétricos respecto a un eje y que en ambos trayectos se siga una velocidad constante con diferente signo. En ese caso, la función apropiada puede ser una función triangular simétrica.

En esta función, el tiempo con derivada positiva es la mitad del periodo, siendo la otra mitad la relativa la derivada negativa. Esto significa que la señal puede ser simétrica respecto a cualquier recta $y = k$ siendo k un número real. En este caso, puede interesar que el eje de simetría sea el eje X, esto significa que, según lo anterior, $k = 0$.

Esta onda se puede obtener integrando una onda cuadrada con ciclo de trabajo del 50 % y simétrica respecto al eje X. Previamente, para obtener una onda cuadra con ese ciclo de trabajo, se puede hallar de la siguiente forma $\dfrac{\sin(\omega t + \varphi)}{|\sin(\omega t + \varphi)|}$.

$$y(t) = \int \frac{\sin(\omega t + \varphi)}{|\sin(\omega t + \varphi)|} \cdot dt = trig(t) \tag{4.90}$$

$$desp = A \cdot trig(t)$$

Al usarse sobre motores que no tiene una respuesta infinitamente rápida, puede darse el caso de que no se alcance el punto objetivo al estar en un punto con una derivada discontinua a cada lado del mismo. Los motores, por los motivos antes descritos, quizás no puedan alcanzar ese punto, quedándose en uno muy cercano. Esto ocasiona un pequeño redondeo en los picos de la señal.

Modificaciones

Como en el caso de la señal sinusoidal, puede interesar mantener al subconjunto perteneciente en la posición objetivo durante más de un instante del tiempo. Por eso, siguiendo el mismo procedimiento que para la señal sinusoidal, se puede obtener la siguiente ecuación.

$$y = \frac{\text{mín}[|trig(\omega t + \varphi)|, val]}{val} \cdot \frac{\sin(\omega t + \varphi)}{|\sin(\omega t + \varphi)|}$$

Este tipo de ecuación presenta un cote simétrico en ambos picos de la señal, provocando que pase más tiempo en los extremos. Además, al estar divido por un escalar menor que la unidad, las pendientes de la señal son mayores que uno.

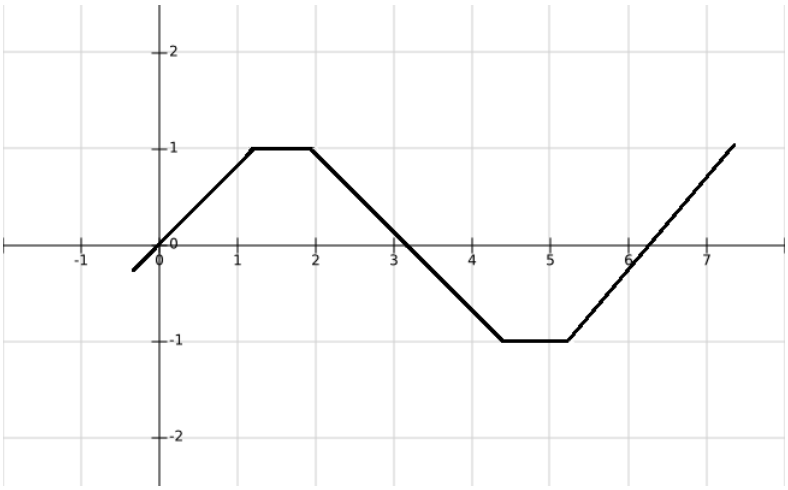


Figura 4.33: Función triangular modificada.

4.4. Desfases para los movimientos de caminata

4.4.1. Introducción

En primer lugar, es necesario definir que desfases hay que aplicar en las trayectorias que corresponden con cada movimiento para sincronizarlas correctamente. La elección de estos valores es algo vital para el proyecto, ya que puede ser necesario reproducir cada movimiento en un determinado instante al mismo tiempo que se realizan otros.

El movimiento de caminata es el encargado de hacer que el robot avance en línea recta tras el paso inicial. Este movimiento es periódico a lo largo del tiempo, y se puede repetir tantas veces como se quiera. Para que se pueda hacer esto, los puntos finales e iniciales del movimiento han de estar próximos entre sí para que no se produzcan discontinuidades entre la ejecución de un ciclo y otro. O, como se ha visto para las trayectorias que se han mostrado, que sigan una misma ecuación cíclica.

Este es el movimiento principal para poder posicionar al robot en el plano al poderlo mover linealmente. Para que el movimiento sea también cíclico, es necesario vincular a los submovientos a funciones que sean periódicas en el tiempo. De esta manera, y siempre que todas tengan el mismo periodo, se podrá generar una movimiento parametrizable que siga una trayectoria predefinida.

Al ser un movimiento que se basa en otros submovimientos, es necesario conocer y fijar los desfases de estos para conseguir un desplazamiento correcto. Si faltase sincronía en estos movimientos, el robot se podría mover en otro sentido o, incluso, llegar a caerse al suelo antes de comenzar a avanzar.

Para lograr relacionar las trayectorias con los movimientos, sólo es necesario multiplicar la señal correspondiente con el movimiento con la amplitud máxima que se quiera alcanzar por cualquiera de las funciones de trayectoria ya vistas del siguiente modo.

$$desp = desp_{max} \cdot f(t)$$

Esta función que depende del tiempo ha de tener el mismo periodo que el resto de funciones temporales que definen el movimiento, para conseguir que todas duren el mismo tiempo y se repitan de manera ordenada, coincidiendo su inicio y final. Además, estas funciones temporales han de tener un elemento que sirva para desfasar unas respecto a otras, por los motivos recién explicados.

4.4.2. Submovimiento de cadera

Sobre este movimiento se definirán el resto de desfases para los siguientes movimientos. Por esto, el desfase de este movimiento ha de ser cero. Además se puede considerar que el primer desplazamiento de la cadera se produce hacia la derecha, y se utiliza este sentido como positivo.

4.4.3. Submovimiento de rodilla

Una vez que se ha decidido que la cadera se considera como referencia para el resto de desfases, el siguiente paso es pensar en como levantar un pie del robot. Para ello, se ha de definir para cada pierna el desfase apropiado, para que el movimiento de rodilla y el movimiento de cadera queden sincronizados. Si se consigue esto, en cada ciclo, el robot podrá separar ambos pies del suelo.

Para que esto se produzca, es necesario definir la trayectoria y desfase para ambas piernas. En primer lugar, se ha de decidir que trayectoria utilizar para alzar el pie. El movimiento de rodilla solo esta definido para desplazamientos positivos. Esto significa que no puede estirar la pierna más de la posición que se define. Para no incumplir esta restricción, se ha de buscar una trayectoria que solo presente valores positivos o nulos.

En este caso, solo es necesario considerar los valores positivos de una de las señales que se acaban de ver, considerando los valores negativos como nulos. En este momento no se va a seleccionar cual de las señales vistas anteriormente se van a usar, ya que eso es materia de capítulo 7.

Suponiendo que se usa una de las trayectorias basadas en funciones sinusoidales, interesa definir los desfases de la señal respecto al movimiento de la cadera. Para el movimiento que se busca, se pretende que sea lo más simétrico posible. Por esto, las piernas han de encogerse con una separación idéntica en el tiempo. Esto significa que han de estar desfasadas medio periodo una respecto a la otra, o lo que es lo mismo, π.

$$\varphi_{rodilla_{der}} = \varphi_{rodilla_{izq}} \pm \pi$$

Para poder levantar un pie, es necesario que el peso del cuerpo del robot esté apoyado sobre el otro. Esto significa que para poder encoger la pierna izquierda, todo el peso ha de estar apoyado en la derecha. Esto significa que la cadera ha de estar en el punto objetivo de amplitud máxima y opuesto al pie a levantar.

Si se considera la trayectoria de la cadera como un seno puro, este punto máximo se alcanza en $\sin(\pi/2) = 1$. Interesa que para este mismo instante, la pierna izquierda se encuentre totalmente encogida, al suponerse en el punto máximo de su trayectoria. Por lo tanto, para que la pierna izquierda se pueda encoger sin soportar peso, y al estar la cadera en el punto máximo a la derecha, ambos desfases han de ser idénticos.

$$\varphi_{cadera} = \varphi_{rodilla_{izq}} = 0$$

Como ya se ha dicho, la pierna derecha ha de ir desfasada πrad respecto a la izquierda para producir un movimiento totalmente simétrico.

$$\varphi_{rodilla_{der}} = \varphi_{rodilla_{izq}} + \pi$$

$$\varphi_{rodilla_{der}} = \pm\pi$$

Si se hiciese el mismo razonamiento para un desplazamiento negativo, hacia la izquierda, se obtendría el mismo desfase para la pierna derecha.

4.4.4. Submovimiento de avance

Si se quiere hacer que el robot avance, es necesario mover hacia adelante y atrás las piernas. Para que este movimiento tenga un sentido práctico, es necesario sincronizarlo con los dos anteriores. Para generar esto, ha de coincidir el avance de la pierna, derivada positiva de la señal, con el encogimiento de la misma pierna. De tal forma que el paso por cero de la señal sinusoidal corresponda con el máximo de la señal de la rodilla.

Esto significa que el desfase con la rodilla, al considerar el máximo una con el paso por cero de la otra, es de $\pi/2$. Para que el movimiento sea correcto, estos valores han de provocar que la pierna se levante instantes antes de que la pierna empiece a avanzar y, del mismo modo, que se estire casi cuando la pierna llega a su posición de avance máxima.

$$\varphi_{avance} = \varphi_{rodilla} + \pi/2$$

Siendo esta la fórmula correspondiente a cada una de la dos piernas.

4.4.5. Submovimiento de cintura

Este movimiento es el que se encarga de rotar la cadera respecto al pecho. Para que este movimiento parezca realista, el giro ha de contraponerse al avance de la pierna, de tal modo que si avanza la pierna derecha, el hombro izquierdo ha de adelantarse. Si se toma como referencia el avance la pierna derecha, este movimiento ha de estar desfasado πrad respecto a esta. Para dejar este desfase referido al movimiento de la cadera, se puede mostrar el desfase como.

$$\varphi_{cintura} = \varphi_{avance_{der}} + \pi$$

$$\varphi_{cintura} = \varphi_{rodilla_{der}} + \pi/2 + \pi$$

$$\varphi_{cintura} = \varphi_{cadera} + \pi + \pi/2 + \pi$$

$$\varphi_{cintura} = +\pi/2$$

4.4.6. Submovimiento de brazos

Para saber el desfase de este movimiento, interesa conocer el desfase respecto a la cadera. Los brazos, intentan avanzar junto a la pierna del lado opuesto. Esto significa que un brazo tiene un desfase respecto al movimiento de la pierna de su mismo lado de π. Esto significa, siguiendo el mismo desarrollo anterior que los brazos están desfasados $\pm\pi$.

$$\varphi_{brazo_{izq}} = \varphi_{avance_{der}} + \pi$$

$$\varphi_{brazo_{izq}} = \pi/2$$

$$\varphi_{brazo_{der}} = -\pi/2$$

4.5. Otros movimientos para mover al robot en el plano

En esta sección se van a ver otras trayectorias diferentes a las vistas hasta ahora para provocar movimientos que permitan colocar y orientar al robot en el plano. En este caso no se hablará de trayectorias simétricas, si no más bien de los movimientos a realizar por el robot para llegar a la posición adecuada para cada caso. Estás trayectorias, en la mayor parte de los casos no son simétricas, ya que interesa que no lo sean para poder cambiar la orientación del robot en el plano o producir un movimiento que al finalizar no devuelva al robot a la posición inicial.

4.5.1. Movimiento de arranque y parada

Para generar una trayectoria en línea recta es necesario dividir el movimiento en otras tres trayectorias más simples. Para evitar que el robot pierda la estabilidad en el primer paso, puede ser interesante empezar a balancear el robot unos instantes antes para que en el momento en el que levante el primer pie del suelo, el otro soporte todo el peso del robot. Además, también puede interesar iniciar el movimiento dando un paso más corto para que la aceleración producida en dirección de avance del robot sea menor al cambiar su velocidad de cero a un valor finito. Si el primer arranque es menor, la fuerza desestabilizadora en dirección de avance también se reducirá.

El caso de querer detener el robot es el mismo que al iniciar el avance. Interesa disminuir de manera progresiva los efectos dinámicos que están presentes en el robot. Para conseguir esto, puede interesar dar un paso más corto para finalizar, y así disminuir la desaceleración del cuerpo al hacerla más progresiva. Además, también puede interesar que el robot no deje de balancearse súbitamente, provocando una aceleración que compense este efecto y que, finalmente, desestabilice al bípedo haciéndolo caer.

Una vez que el robot halla iniciado la caminata y se mantenga andando en línea recta hasta que se decida pararlo, el robot podrá ejecutar tantas veces como se desee la trayectoria correspondiente a un solo periodo de la caminata. Este periodo, que ya ha sido tratado en profundidad en los capítulos anteriores, consistirá en dar dos pasos, uno con cada pierna, mientras se balancea para conseguir desplazar el peso del robot.

Estos movimientos son los encargados de iniciar y finalizar el movimiento de de caminata. El primero de ellos, el movimiento de arranque, precede al movimiento principal para iniciar el balanceo instantes antes de que comience el robot a avanzar. Además de este balanceo, interesa que el primer paso sea algo más pequeño o que solo se mueva una de las dos piernas, quedando el cuerpo del robot en una posición tal que no se desestabilice el robot cuando se ejecute por primera vez el movimiento de caminata.

Por otro lado está el movimiento de parada, que se encarga de hacer que el robot se detenga de forma controlada tras un movimiento de caminata. Para detener el robot, es necesario seguir los mismos pasos que el movimiento de arranque pero en sentido opuesto. Primero es necesario detener el avance del robot de forma progresiva, al dar los último pasos más cortos y, después, ir disminuyendo el balanceo del robot hasta el reposo. Estos dos movimientos puede solaparse dependiendo de la estabilidad del robot.

Movimiento de arranque

Basándose en lo visto hasta ahora, se puede usar el movimiento de caminata como punto de partida e incluir ciertas modificaciones para adaptarlo.

En primer lugar, si se desea balancear el robot antes de iniciar la marcha, se puede generar una trayectoria que parta del reposo y alcance el valor de amplitud que se vaya a usar para el siguiente movimiento de caminata. Para ello, solo es necesario usar el submovimento respectivo

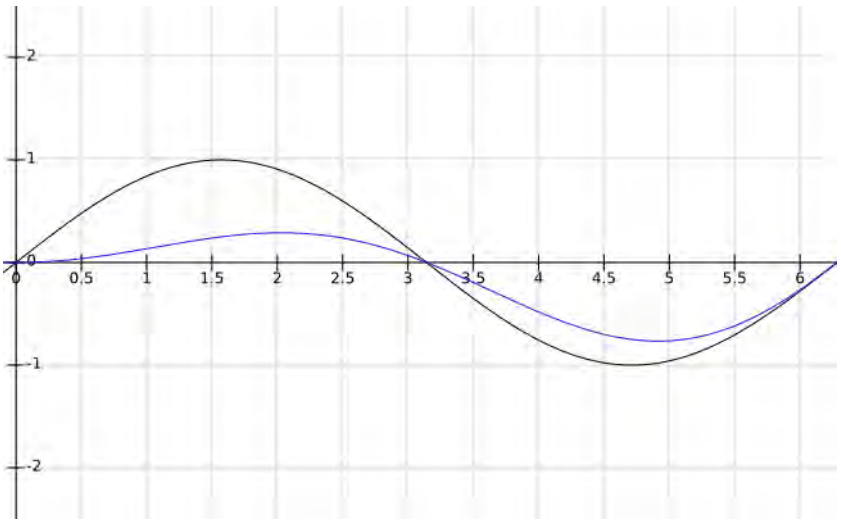


Figura 4.34: Amplificación de la función por una variable escalar.

a la cadera, dejando los demás parámetros de control en cero. Para lograr este inicio se pude hacer de diferentes maneras, pudiendo elegir la que más convenga, pero en este caso solo se analizarán dos de ellas.

Balanceo. Variable escalar La primera consiste en utilizar una variable escalar que multiplique al parámetro de control. Esta variable, puede aumentar siguiendo cualquier tipo de progresión ascendente entre cero y uno, pero por simplificar solo se analizará este aumento como una función lineal.

$$desp = desp_{max} \cdot f(t) \cdot e(t)$$

Interesa que el periodo de la variable coincida con el de la caminata o que sea múltiplo del mismo. Esto, si no se tiene en cuenta, puede producir una discontinuidad entre ambos movimientos, pudiendo provocar una desestabilización del robot.

$$T_e = n \cdot T_{caminata} \qquad n \in \mathbb{N}$$

Además, se puede elegir utilizar el semiperíodo como la base de tiempo de la variable. Si se elige esta opción, hay que tener en cuenta si n es positiva o negativa. Esto significa que si es impar, el movimiento de inicio acabará en el lado opuesto que el inicio del movimiento de caminata, generando una discontinuidad. Para solucionar esto, se puede invertir el signo de la función. Para hacerlo de un modo genérico e incluir estas opciones en una sola, se puede expresar como lo siguiente.

$$desp = desp_{max} \cdot f(t) \cdot \left[(-1)^n \cdot e_{\frac{T}{2}}(t)\right] \qquad n \in \mathbb{N}$$

Como se ha dicho, interesa que la función $e(t)$ sea lineal. Para ello, si se considera que el tiempo también es lineal, se puede definir esta función como.

$$e_T(t) = \frac{t}{n \cdot T} \qquad e_{\frac{T}{2}} = \frac{t}{n \cdot T} \qquad 0 \leq t \leq n \cdot T$$

Para hacer un movimiento que no sea excesivamente lento, no interesa que se tarde mucho tiempo en empezar a balancear el robot. Por este motivo, es recomendable elegir valores pequeños para n, por ejemplo 1 o 2.

Balanceo. Concatenación de movimientos La segunda forma de iniciar el balanceo del robot consiste en unir movimientos que tengan un período igual al de la caminata y que la amplitud de estos vayan aumentando progresivamente desde cero hasta el valor de la amplitud del movimiento final.

Si se ha utilizado como referencia de desfase el movimiento de cadera, todos los cambios del valor máximo de la amplitud se producen por uno de los dos pasos por cero de la función sinusoidal. Esto significa que solo aumenta bruscamente la velocidad lineal de la cadera en este punto. Se puede considerar que ese punto es el más estable de todas las posiciones del robot al proyectar el centro de gravedad a igual distancia del apoyo de los pies, por lo que, aunque se aumente repentinamente el valor de la velocidad en ese punto, los esfuerzos laterales que produce la aceleración se absorben rápidamente.

El principal inconveniente de este sistema de inicio de balanceo es que puede llegar a ser lento, al tener que concatenar varios movimientos de un mismo periodo.

Inicio de avance Una vez que el robot ha empezado a balancearse para repartir el peso, se puede empezar a mover la pierna en la que no esté apoyado el peso del cuerpo. Para no provocar una aceleración brusca, interesa que el primer paso del robot sea más pequeño, pudiendo llegar a mover solo una de las dos piernas. Este paso de inicio se ha de dar con la pierna opuesta con la que el movimiento de caminata mueva primero, para que al iniciar este último el robot esté ya en una posición similar a la que se produciría cuando acabase un período de caminata. Esto, además origina que en los tres primeros cuartos del movimiento de inicio el robot solo se mantenga balanceándose.

En el último cuarto de período se produce el movimiento de la pierna como tal. En esta fracción, el robot ha de añadir, además del de balanceo, el movimiento de avance y el de rodilla.

En primer lugar, la trayectoria del movimiento de avance se puede definir del mismo modo que lo hecho en la caminata, salvo que solo interesa el último cuarto. Esto ocasiona que salvo en dicho espacio de tiempo, el desplazamiento de ese movimiento es nulo. En cambio, en el período indicado el desplazamiento aumenta linealmente desde el reposo hasta la amplitud objetivo. Hay que tener en cuenta que esta señal triangular ha de tener la misma amplitud y valor umbral que la homóloga del movimiento de caminata.

Por otro lado, la trayectoria del movimiento de la rodilla no se puede definir del mismo modo, pero si de uno similar. Al estar este movimiento retrasado con un desfase de π, al inicio del último cuarto del período se alcanzaría el valor máximo de la trayectoria, por lo que se produciría una discontinuidad. Para evitar esto se pude desplazar el valor máximo unos instantes, utilizando un valor de desfase ligeramente mayor. Además de hacer esto, hay que tener en cuenta que el valor de esta nueva señal ha de ser cero en el instante en el que la trayectoria triangular alcanza su valor objetivo. Para evitar esto, se puede aumentar el número de exponentes del seno. Por último, por ser un movimiento corto en el tiempo, puede interesar definir una amplitud menor que la de la caminata para lograr que a ningún motor se le pida más velocidad de la que puede dar.

Movimiento de parada

El movimiento de parada es el encargado de llevar al robot a la posición de reposo partiendo de un movimiento de caminata.

Este movimiento se puede considerar como el complementario del movimiento de arranque, al realizar el movimiento puesto. En este caso interesa hacer que el robot deje de sufrir esfuerzos dinámicos de forma progresiva.

Concrétamente, este movimiento, partiendo de el movimiento de caminata, ha de detener el avance del robot dando un paso más corto o que diferente para finalizar el avance y, posteriormente dejar de balancear el centro de gravedad del robot hasta llevarlo al reposo.

Como se ha detallado anteriormente el movimiento de arranque, este no se analizará en profundidad, solo dando las claves necesarias para realizarlo.

En este caso, el movimiento para detener el avance se produce en el primer cuarto del período. En este caso, la pierna que se mueve es la que inicia el movimiento de caminata y parte de la región de valores negativos. Por lo tanto, el movimiento lineal ha de producirse desde el punto de máxima amplitud negativo hasta cero. El caso de la rodilla es comparable, al ser la trayectoria similar a su homóloga del movimiento de arranque, salvo que se produce el movimiento en el primer período.

Para cesar de balancear el robot, se puede recurrir a las dos formas de hacerlo antes detalladas. En el caso de usar la concatenación de movimientos, las amplitudes han de ser menores que la del movimiento de caminata, disminuyendo progresivamente en cada ciclo hasta hacerse nula. Si se usa la variable escalar hay que seguir las mismas fórmulas, salvo que la variable en el momento inicial ha de ser uno y al acabar el período ha de ser nula. Por lo tanto, esto se puede expresar del siguiente modo.

$$e_T(t) = 1 - \frac{t}{n \cdot T} \qquad e_{\frac{T}{2}} = 1 - \frac{t}{n \cdot T} \qquad 0 \leq t \leq n \cdot T$$

4.5.2. Movimiento de giro

Este movimiento es el encargado de hacer que el robot rote sobre sí mismo para poder cambiar su orientación sobre el plano. Este movimiento, a diferencia que los anteriores no es periódico aunque la posición inicial y final de los motores tras este movimiento sí lo sean.

Para poder conseguir girar el robot, es necesario repartir el peso del robot sobre una pierna, alzar la otra mínimamente del suelo para no perder estabilidad, cambiar la orientación de esa pierna al girarla hacia el exterior del cuerpo, apoyar de nuevo el pie en el suelo, balancear el paso sobre la otra pierna para poder alzar la primera y, por último, hacer descender el pie al mismo tiempo que se rota de nuevo la pierna que se ha orientado previamente para devolverla a su posición inicial. En el último movimiento, al volver a girar la pierna respecto a la cadera, se reorienta todo el cuerpo del robot por estar apoyado sobre dicha pierna. Por lo tanto, el ángulo girado por el robot es la amplitud del ángulo que gira el eje q_1, al ser este el que orienta la pierna.

Este movimiento se puede realizar para girar al robot en sentido horario o antihorario. Para cada uno de estos sentidos de giro se ha de crear una cinemática diferente, que use primero una pierna u otra en función del movimiento a realizar. Estos dos movimientos han de ser simétricos respecto al plano de perfil para que el ángulo girado en uno u otro sentido sea el mismo.

El movimiento generado hace girar al robot un ángulo determinado. Si se quiere girar al robot de puede invocar a este movimiento un número determinado de veces para que complete esa orientación total. No se recomienda giros excesivamente grandes, ya pueden desestabilizar al robot. Por eso se aconseja usar repetidas veces un giro menor hasta completar el giro deseado.

Como se acaba de ver, este movimiento no responde a una señal simétrica ni periódica vistas en la sección 4.3, por lo que hay que definir y detallar cada señal sin apoyarse en las trayectorias ya definidas. Por simplicidad, se puede considerar que todos los cambios se hacen a velocidad lineal constante y se mantienen, en caso de ser necesario, en la posición final hasta que se le vuelva a dar una orden a cada conjunto de motores.

Movimiento de giro alternativo

Se puede simplificar enormemente el movimiento de giro si se considera que los pies del robot pueden deslizar sobre el suelo y se aprovechan los esfuerzos dinámicos para rotar el robot. Como se ha mostrado anteriormente en 2.2.1, el robot dispone de un grado de libertad entre la cadera y el pecho que permite el giro relativo entre ambos. La masa presente en la mitad superior del robot es significativa, por lo que al rotarla sobre este eje a cierta velocidad puede producir una aceleración inercial sobre el robot. Esta aceleración depende de la velocidad y recorrido que se desee imprimir sobre el motor.

Apoyándose en esto, se puede deslizar los pies del robot usando la inercia. Para producir un cambio en la orientación, es necesario generar dos movimientos a velocidades distintas. Uno de los dos movimiento ha de mover al robot desde la posición actual al inicio de este movimiento mientras que el segundo ha de ejecutarse a una velocidad alta al moverse desde la posición inicial hasta la final. Este segundo desplazamiento es realmente el que produce el giro relativo, por lo que es necesario dimensionarlo correctamente. Es importante resaltar que una velocidad ha de ser muy superior a la otra para producir el efecto deseado al provocar aceleraciones diferentes.

Este movimiento tiene importantes desventajas aunque conceptualmente sea muy sencillo. El primero es que es necesario que los se pies deslicen sobre la superficie del suelo. Si el deslizamiento es muy elevado, las reacciones que han de producirse en el movimiento de avance se harían casi nulas, no pudiendo generar el avance del robot. Por otro lado, si la adherencia es alta, puede provocar que el robot no gire o se desestabilice.

Además, el giro no está totalmente controlado, al depender de la adherencia de los pies, la inercia total del robot, de la inercia de la parte superior del robot que a su vez dependen de la posición de los brazos, y del recorrido del giro y su velocidad. Esto significa que es difícil definir el giro exacto del robot, al no poder prever el coeficiente de rozamiento para cada tipo de suelo.

Por esto, si se desease usar esta simplificación, al no saber exactamente cual es el giro provocado, es casi necesario utilizar algún tipo de sensor que te indique el giro relativo que se ha producido respecto a la posición anterior. Por citar algún tipo de sensor para este propósito, se puede usar brújulas electrónicas o giroscopios que indique en ángulo girado en función del campo magnético de la tierra o de la velocidad angular presente en el cuerpo en los instantes del movimiento.

4.5.3. Movimiento de desplazamiento lateral

El movimiento de desplazamiento lateral es el encargado de hacer que el robot se desplace en los sentidos perpendiculares al de caminata. Este movimiento hace que el robot se desplace en línea recta pero lateralmente sin cambiar su orientación respecto al plano. Esto puede ser muy útil a la hora de esquivar obstáculos que estén situando de frente, pudiendo sortearlos sin cambiar la orientación del movimiento.

Este movimiento, al igual que el anterior, se puede realizar en uno u otro sentido.

Para realizar este desplazamiento, el robot ha empezar teniendo apoyado todo su peso en una de sus piernas, por eso es necesario empezar desplazando la cadera. Una vez hecho esto, se ha de levantar el pie opuesto del suelo una vez que deje de soportar peso. Tras esto, el extremo de la pierna ha de desplazarse hacia el exterior, estirando esta al llegar al punto final. En este punto, el robot ha de balancear el peso para que la pierna que está en el aire se pose en el suelo, teniendo en cuenta que el pie ha de apoyarse de forma plana. Una vez que los dos pies estén en el suelo, es necesario continuar el desplazamiento de la masa del robot para que toda ella quede apoyado sobre la otra pierna. Tras esto, la pierna que ha quedado libre de peso se podrá alzar, y recolocarla para que quede en una posición de balanceo propia del movimiento de caminada.

Por último, partiendo de dicha posición de balanceo, es necesario volver a centrar el centro de masas del robot al devolverlo a su posición inicial.

Para poder alejar el extremo del robot de forma controlada, se puede usar un estudio de movimiento de avance visto en 4.2.3, ya es similar a este al tener la pierna totalmente estirada y desplazar el extremo por un plano. El único cambio que hay que tener en cuenta es que la distancia de la longitud total de la pierna es diferente, al ser la distancia existente entre los ejes q_2 y q_7.

De modo similar al anterior, las trayectorias no son cíclicas ni responden a ninguna trayectoria vistas en el capítulo 4.3, por lo que es necesario definirlas manualmente. Para simplificar el cálculo, se han de considerar que los desplazamiento entre puntos de una misma cinemática se realizan a velocidad constante.

Capítulo 5

Selección de componentes

En este punto se pretende diseñar y definir los elementos que componen el sistema eléctrico del robot. En primer lugar, conociendo los servomotores que usa el robot, se pretende diseñar el subcircuito de alimentación que permita dotar de energía al robot. A continuación, es necesario definir cual es la placa que hará las veces de controladora del robot. Por último, se muestra como queda el circuito eléctrico del robot y el montaje del mismo a nivel físico.

5.1. Alimentación de los motores

Una vez decididos los elementos motores es necesario pensar en como alimentarlos y controlarlos. Con la idea de hacer un robot autónomo, la alimentación de los motores ha de incluirse a bordo del prototipo. Como se ha visto antes, para alimentar los motores necesita un voltaje entre los 5 y 6 voltios. Además, si se suman todas las intensidades demandadas por cada uno de los motores superan los 3 amperios, y 10 amperios el caso de que todos los motores estén en un estado transitorio. Por esto, se necesita una fuente de energía a bordo del robot que suministre un voltaje constante soportando cambios repentinos de corriente.

5.1.1. Batería

En primer lugar, para almacenar la energía se recurre a una batería LiPo. Este tipo de baterías se caracterizan por no tener un peso elevado, además que permiten almacenar una gran cantidad de energía en ellas. Dependiendo del modelo, estas baterías permiten tasas de descargas varias veces superior a su capacidad total sin sufrir daños o deteriorarse. Estas características las hacen idóneas como fuente de alimentación para nuestro dispositivo portátil al tener un peso bajo en comparación con otro tipo de baterías, como las de plomo o níquel por ejemplo, soportando descargas bruscas sin perder vida útil o prestaciones y permitiendo almacenar en un espacio reducido la energía suficiente para mover al robot durante varios minutos.

Aunque las baterías de níquel presenten una relación capacidad/volumen que las baterías LiPo, tienen una peor relación en cuanto a capacidad/peso al ser más densas. Por otro lado, las baterías de plomo están destinada a poder resistir descargas muy elevadas, teniendo mayores tasas de descarga que la LiPo. Del mismo modo, la densidad de estas baterías es alta en comparación con las baterías LiPo. En este caso, se pretende optimizar la relación de capacidad/peso en el robot, al intentar aligerar el peso total del robot. Por este motivo se pretende utilizar una batería LiPo para este proyecto.

La pila eléctrica de este material ofrece 3.7 voltios nominales y 4.2 voltios cuando está totalmente cargada, por lo que es necesario agrupar en serie varias de estas celdas para elevar el voltaje. En este caso, se necesita tener un voltaje constate en el tiempo y de un valor cercano a los 6 voltios, por lo que se ha de recurrir a otro elemento para regular este voltaje.

Por lo tanto, se elegirá una batería LiPo de tres células, sin descartar el uso de una de dos, ofreciendo un voltaje mayor al deseado y previendo en uso de un regulador que estabilice la tensión. Estas batería disponen de diferentes cargas, dependiendo se sus dimensiones al estar relacionado con parámetros geométricos. Al disponen espacio suficiente en la parte trasera del robot, se pueden usar baterías de este tipo de hasta 3000mAh. No obstante, cuanto mayor es la batería mayor es su precio, por lo que se ha optado por adquirir para el robot una batería de 1750mAh de la marca *Rhino*, con una tasa de descarga instantánea de hasta veinte veces su capacidad, que se puede ver en la figura 5.1.

El lugar donde está ubicada la batería en el robot es la parte trasera de la cintura, ya que no entorpece la movilidad de ningún eslabón u otro elemento, elevando lo mínimo el centro de gravedad del robot. Por simplificar el sistema eléctrico del robot, la batería deberá cargarse de forma externa al robot en un cargador especializado para ese propósito.

Figura 5.1: Batería Rhino con tres células y 1750mAh de capacidad.

5.1.2. Regulador

Para el correcto funcionamiento de los motores se necesita tener un voltaje constante dentro de los rangos de funcionamiento de estos. El voltaje de la batería decae según se descarga, no siendo constante en el tiempo. Otro gran inconveniente es que no se pueden sacar directamente los 6 voltios que se necesitarán, ya que solo se pueden conseguir voltajes múltiplos de 3,7 voltios, al conectar en serie varias de estas células.

Por esto es necesario colocar un regulador de voltaje entre la batería y los motores para conseguir una tensión estable. Dentro de los reguladores existen varios tipos, que presentan diferentes características y rendimientos, que se detallarán a continuación.

Reguladores de voltaje lineales

Los reguladores de voltaje lineales son componentes electrónicos que al alimentarlos a una tensión superior a la requerida, suministran un voltaje nominal de salida constante. Estos reguladores presentan rendimientos entorno al 60 % y disipan la energía sobrante en forma de calor, por lo que se calientan fácilmente y, en ocasiones, es necesario usar disipadores de calor.

Dentro de este tipo de reguladores está la familia de componentes $LM78XX$, que ofrecen una gama de diferentes voltajes en función del valor de los dos últimos dígitos, representados por las dos X. En la figura 5.2 se puede ver un modelo que ofrece 5 voltios a la salida. Estos componentes se caracterizan por un bajo coste y unas dimensiones de unos pocos milímetros, estando muy extendidos en numerosas aplicaciones. Esta familia de componentes electrónicos soportan corrientes de salida de hasta 500 miliamperios, quedándose muy por debajo de la corriente requerida por el robot.

Figura 5.2: Regulador lineal LM7805.

Existen otras gamas de componentes con rendimientos y cargas de trabajo superiores a los LM78XX que admiten hasta 2 amperios en la salida, pero siguen siendo insuficientes para la correcta alimentación del robot.

Para poder usar reguladores lineales dentro del proyecto, sería necesario hacer divisiones del circuito eléctrico y alimentar cada una de estas subdivisiones de forma independiente con uno de estos reguladores. Cada una de estas subdivisones no puede consumir más potencia de la que puede dar por separado cada uno de estos elementos. Para el correcto funcionamiento eléctrico, todas las referencias a masa de todos estos circuitos secundarios debe ser la misma, para evitar problemas de comunicación con los elementos o cualquier otro tipo de inconveniente. Desgraciadamente, viendo el consumo de los motores, se podrían agrupar un número pequeño de elementos en cada subconjunto, haciendo muy complejo es esquema eléctrico. Si además atendemos a que tiene un rendimiento bajo, es casi necesario descartar este tipo de regulador para la implementación en este proyecto.

Reguladores de voltaje conmutados

Los reguladores de voltaje conmutados son circuitos electrónicos que permiten o no el paso de la corriente a través de un interruptor a una frecuencia elevada, permitiendo la carga y descarga de componentes pasivos, produciendo un voltaje constante diferente al de entrada.

Estos elementos tienen un rendimiento cercano al 95 % al estar construidos principalmente con bobinas y condensadores, que no consumen energía. Dependiendo de la colocación de los componentes, existen diferentes tipos de reguladores conmutados, pudiendo elevar, reducir o elevar y reducir el voltaje a la salida con respecto a la entrada. En este caso, solo se detallarán los reguladores de tensión reductores, como el que se puede ver en la figura 5.4, ya que usaremos una batería con dos o tres células de 3.7 voltios, requiriendo únicamente $6V$ constantes en el tiempo.

A parte de la principal ventaja de rendimiento de estos reguladores, se puede dimensionar la potencia del circuito como más convenga, no habiendo un límite estricto al no tener que prever el sobrecalentamiento que producen las pérdidas en los semiconductores en forma de calor. Además, estas fuentes de alimentación suelen llevar integradas un lazo de control, que monitorizan el voltaje en cada instante del tiempo, cambiando su respuesta en el caso de aumente la corriente o la tensión de salida repentinamente, reajustándose para dar siempre el mismo valor de tensión.

Este tipo de reguladores tiene de forma genérica unos componentes específicos, que son una bobina o transformador, un transistor, un diodo, un condensador y el lazo de control que genera la señal de activación del transistor. El transistor hace las veces de interruptor, permitiendo el paso de la corriente a través de él a una alta frecuencia. La señal que habilita el transistor según se necesite es una onda PWM (*modulación por ancho de pulso o Pulse Width Modulation*). En la figura 5.3 se puede ver un ejemplo de varios ciclos de trabajo a una misma frecuencia.

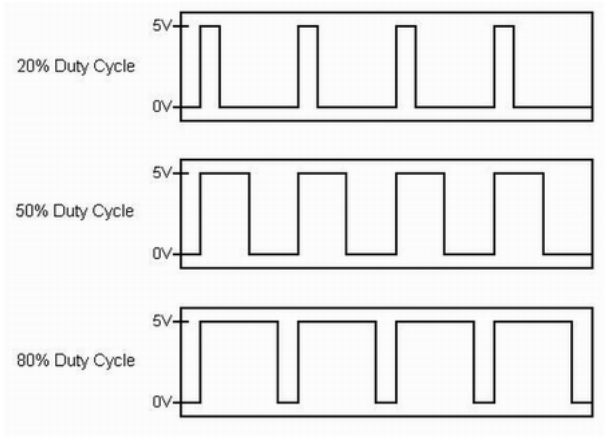


Figura 5.3: Diferentes ciclos de trabajo para una onda PWM.

Esta onda periódica varía el ancho de pulso a nivel alto, o ciclo de trabajo, sin modificar el periodo. Al variar el ciclo de trabajo se puede cambiar el tiempo que permanece abierto el transistor, dejando pasar más o menos potencia a la salida del regulador.

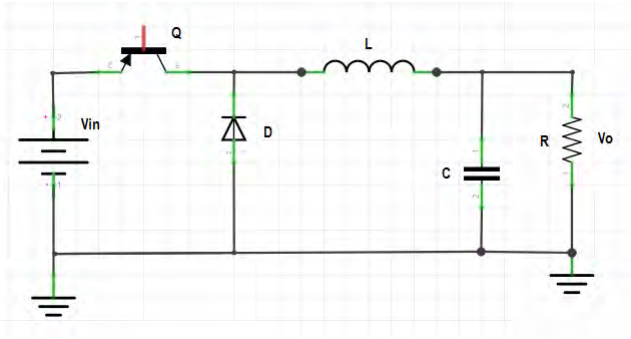


Figura 5.4: Convertidor de tensión reductor.

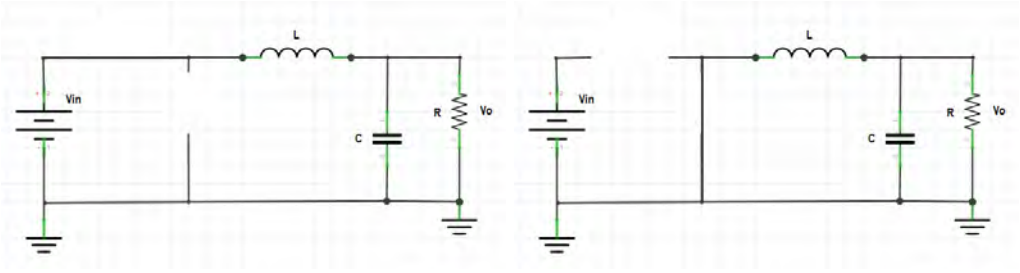
Con idea de simplificar el cálculo, se han de considerar que todos los componentes son ideales. Estos quiere decir que, salvo la resistencia R que representa a la carga, ningún componente tiene resistencia interna, por lo que no se producen ni pérdidas ni caídas de tensión en ninguno de los elementos. Para analizar este circuito y saber como se comporta la salida de este según la entrada es necesario saber que para que una bobina se encuentre en estado estacionario, el voltaje medio sobre esta ha de ser cero. Del mismo modo que para que un condensador se encuentre en estado estacionario la corriente media a través de el ha de ser nula.

$$\bar{v_l} = 0 \tag{5.1}$$

$$\bar{i_c} = 0 \tag{5.2}$$

En el caso, de como se ha dicho antes, el transistor podrá estar dejando pasar corriente o no. Por lo tanto es necesario analizar este mismo circuito como dos diferenciados como se ve en la figura 5.5, estando por un lado el circuito perteneciente a los tiempos t_{ON} en los que se circula corriente a través del transistor y por otro cuando no t_{OFF}. si, como hemos dicho antes, el circuito de abre y se cierra periódicamente, el la suma de estos tiempo ha de coincidir con el periodo $T = t_{ON} + t_{OFF}$. [1]

[1] El análisis de este circuito solo responde a un simple estudio para introducir al lector levemente en los convertidores de DC, no siendo un estudio pormenorizado de todas las características y detalles que abarca este campo. La simplificación que se hace en $T = t_{ON} + t_{OFF}$ responde a un estudio que se imponen las condiciones de contorno de que el circuito ha de estar en estado estacionario y que funciona el convertidor funciona en *Modo de Conducción Continuo*. Esto último significa que solo existen dos estados en los que el circuito puede encontrarse, siendo t_{ON} y t_{OFF}. Esto se debe a que



(a) Estado T_{ON} (b) Estado T_{OFF}

Figura 5.5: Estados T_{ON} y T_{OFF} del circuito regulador.

Para saber como se relaciona el voltaje de la entrada con el de la salida es necesario atender a la ecuación 5.1. Para ello, es necesario relacionar el voltaje que se ocasiona en en cada uno de los dos estados y multiplicarlo proporcionalmente por el tiempo en el que se encuentra en ese estado para ponderarlo.

En primer lugar se ha de analizar el circuito en el estado correspondiente a t_{ON}. En este caso, el transistor se encuentra cerrado y dejando pasar corriente a través de él. Como se puede ver, el diodo queda con un voltaje mayor en el cátodo que en el ánodo, quedando en inversa, no permitiendo el paso de corriente a través de él. Para mostrar esto, se considera al transistor como un cortocircuito y al diodo como un circuito abierto.

Si se aplica la segunda ley de Kirchhoff, *en un lazo cerrado la suma de todas las caídas de tensión es igual a la tensión total suministrada. De forma equivalente, la suma algebraica de las diferencias de potencial eléctrico en un lazo es igual a cero.*

$$\sum_{i=1}^{n} v_i = v_1 + v_2 + v_3 ... + v_n$$

$$\sum_{i=1}^{n} v_i = 0$$

Si se considera al circuito como un lazo, se puede ver que

$$v_i - v_l - vo = 0$$

$$v_l = v_i - v_o \tag{5.3}$$

Por otro lado, si se analiza el circuito en el estado provocado por t_{OFF}, el transistor no permite pasar corriente a través de él. Al existir una bobina en serie, no se puede interrumpir la bruscamente la corriente que pasa por ella. En el caso de abrir el transistor en cuando existe corriente pasado por la bobina, se crea una diferencia de voltaje proporcional a la derivada de la corriente. Por se un cambio finito en un período de tiempo infinitesimal, se provoca un voltaje muy elevado. Este voltaje provocaría la ruptura dieléctrica del material del transistor al producir un arco eléctrico para oponerse al cambio de intensidad. Para evitar que la corriente por la bobina sufra un cambio brusco, se coloca en antiparalelo con la carga al diodo, para que exista una vía por donde la corriente pueda circular cuando el transistor esté abierto. Por eso, se considera para este subcircuito al transistor como un circuito abierto y al diodo como un cortocircuito.

Si se aplica en el lazo de la carga la segunda ley de Kirchhoff, se obtiene

$$-v_l - v_o = 0$$

la intensidad en la bobina nunca llega a hacerse nula en ningún momento del ciclo, dicho de otro modo se cumple que $\frac{1}{2}\Delta i_l \leq \bar{i}_l$. Si en algún momento esta corriente es cero, o $\frac{1}{2}\Delta i_l \geq \bar{i}_l$, el circuito tendrá un tercer estado en cada periodo $T_{discontinuo}$ y se corresponderá con un modo de funcionamiento llamado *Modo de Conducción Discontinuo.*

$$v_l = -v_o \tag{5.4}$$

Como se ha dicho anteriormente, para que el circuito esté en estacionario se debe cumplir 5.1. Por lo que si aplicamos lo obtenido en 5.3 y 5.4 se puede plantear

$$v_{l_{tOFF}} \frac{t_{ON}}{T} + v_{l_{tOFF}} \frac{t_{OFF}}{T} = 0$$

Si se considera $\frac{t_{ON}}{T}$ como el ciclo de trabajo D.

$$(v_i - v_o)D + (-vo)(1 - D) = 0$$

$$v_o = v_i \cdot D \tag{5.5}$$

Se puede ver fácilmente que el el valor de voltaje de salida es proporcional al de entrada por un factor multiplicativo que puede tomar valores entre cero y uno.

Una explicación breve e intuitiva de este tipo de circuito es que la bobina y el condensador actúan como un filtro paso-bajo, eliminando todos los armónicos de la señal PWM salvo la componente continua. El valor de dicha componente es proporcional al tiempo que el transistor deja pasar la corriente en cada ciclo, siguiendo la fórmula 5.5.

Otra ventaja que presenta este tipo de regulador reductor es que demanda una tasa de descarga menor, si se compara con extraer la energía directamente o con un regulador lineal. Si aplicamos un balance de potencias en la salida y la entrada del reductor obtenemos.

$$Po + Px = Pi \tag{5.6}$$

Siendo Po la potencia a la salida, Pi la potencia a la entrada y Px las pérdidas que se producen en el reductor. Si se desglosa la potencia en voltaje e intensidad vemos

$$Vo \cdot Io + Px = Vi \cdot Ii \tag{5.7}$$

Como el voltaje a la entrada y la salida están relacionados por la fórmula 5.5, se obtiene

$$Vi \cdot D \cdot Io + Px = Vi \cdot Ii \tag{5.8}$$

Despejando la corriente de entrada Ii, podemos ver la siguiente ecuación

$$Ii = \frac{Px}{Vi} + D \cdot Io \tag{5.9}$$

Como se puede ver en la fórmula 5.9, y teniendo presente que el ciclo de trabajo puede tener valores entre 0 y 1 y que las pérdidas son muy pequeñas por tener un rendimiento elevado, la corriente que se demanda a la batería es menor que la de salida. Esto produce una tasa de descarga menor en la batería, forzando en menor medida las células, y pudiendo tener tandas de uso de la batería de mayor duración usando la misma carga.

Como desventaja, de los reguladores conmutados presentan un rizado en la salida. El filtro generado por la bobina y el condensador no es ideal, dejando pasar algo de ruido. Estas interferencias pueden presentar inconvenientes en circuitos muy precisos, no siendo el caso de los motores del robot que también generan ruido electromagnético cuando están en movimiento.

UBEC Turnigy 8A

Se puede entender que se elija un reductor conmutado para montarlo en el robot por los motivos antes citados. En concreto, y para poder dejar una solución comercial a quien desee realizar una copia de este robot, se ha elegido como regulador de tensión el modelo *UBEC Turnigy 8A*.

El modelo *UBEC Turnigy 8A* es un regulador de tensión conmutado pensado para aeromodelismo, donde también aparece el mismo problema de alimentación. Este modelo pertenece al grupo de los reguladores de tensión reductores, o *Buck*, con lazo cerrado de control. La corriente nominal máxima que permite este regulador es de 8 amperios, pudiendo alcanzar picos de 15 amperios durante un breve periodo de tiempo. El voltaje de salida de este regulador es seleccionable mediante un interruptor, pudiendo elegir entre una salida de 5 o 6 voltios. Una particularidad de este modelo es que viene diseñado para baterías LiPo de 2 y 3 células, por lo que tiene 4 indicadores *LED* que muestran el estado de la carga restante en la batería. Al estar diseñado para aeromodelismo, presenta un tamaño compacto de 42x39x9mm y peso de 34g, además de incluir un blindaje magnético para evitar interferencias con el ruido de cualquier otro elemento a bordo.

Figura 5.6: UBEC Turnigy 8A.

El lugar donde esta alojado este elemento dentro del robot es la parte delantera del pecho, estando suficientemente cerca de la batería para usar la menor longitud de cable posible.

5.2. Controladora

Para poder manejar los motores es necesario un elemento capaz de controlar todos los motores de forma simultanea y a tiempo real, del mismo modo que se pueda ampliar los elementos conectados pudiendo poner nuevos sensores o actuadores. Con idea de que otras personas puedan seguir desarrollando este robot por su cuenta, es aconsejable usar un elemento fácil de manejar y de conseguir. Además, para poder tener cierta libertad y flexibilidad en las tareas que pueda hacer el robot, es indispensable que dicho elemento sea reprogramable.

De las placas vistas anteriormente en el apartado 3.4, es necesario seleccionar cual es la más adecuada. La característica principal que han de tener es que sean capaces de controlar a tiempo real el robot mediante señales PWM. Para evitar tener que recurrir a placas intermedias, es necesario que la placa sea capaz al menos de controlar 24 señales, pudiendo adaptarse a la frecuencia de controla de los servos y poder ajustar su ciclo de trabajo según se demande.

Muchas de las placas vistas hasta ahora incluyen conectores que no se usan, como salidas de vídeo y audio. Al no ser una necesidad para el proyecto, estos conectores no se deben valorar, siendo en ocasiones elementos que pueden molestar a la hora de incluir una placa de expansión o que desequilibre en centro de masas del robot.

Otro dato importante a tener en cuenta es el precio de la placa. Si se pretende realizar un robot de bajo coste, es necesario que no existan componentes con costes relativos superiores a otros elementos. Al utilizar motores de bajo costo, no es lógico invertir en una controladora con

un precio elevado, y más si sus capacidades van a verse capadas por las prestaciones de otros elementos más baratos y con menores prestaciones. Por este motivo es necesario tener especial atención del precio final de la placa.

Por último, es necesario valorar las capacidades de la placa, en especial la velocidad de procesamiento de datos. La placa ha de ser capaz, como mínimo, de poder gestionar los motores a tiempo real y tener capacidad, en previsión de futuro, de poder comunicarse con sensores y otros elementos.

Por lo tanto, según estos motivos se puede considerar a la placa *Arduino Mega 2560* como la más adecuada para este propósito. Esta placa presenta los recursos suficiente para la gestión de el elevado número de motores de los que se dispone, tanto por el número de pines disponibles como por los TAUs presentes en la placa que permiten generar ondas PWM para el control de los servos. Varias de las placas vistas, no presentan tantos pines o TAUs, como *Rashberry Pi B o Intel Galieo*, descartándolas inmediatamente por no se capaces de cumplir el requisito mínimo. Por otro lado, la placa *RoBoard RB-110* presenta 16 patillas por las que generar señales PWM. No obstante, se pueden sacrificar recursos de comunicaciones para soportar más servomotores.

Aunque la velocidad de procesamiento de la placa elegida sea realmente baja en comparación de la *Beaggle Bound Black* o *RoBoard RB-110*, la elección final ha sido tomada por el precio. La placa *RoBoard RB-110* tiene un coste desproporcionado para este proyecto al tener un precio superior al coste de los servomotores. Por otro lado, la placa *Beaggle Bound Black* presenta un coste similar a la placa *Arduino Mega 2560* original. No obstante, como se ha dicho antes, existen réplicas idénticas por la cuarta parte del precio de la original. Aunque se descarte la placa *Beaggle Bound Black* para este proyecto, sigue siendo una opción excelente para un robot de estas características. Al ser este proyecto relativo exclusivamente a la movilidad del prototipo, esta placa queda en segundo lugar. Pero si se analizase en un futuro la idea de integrar visión o cualquier otro sistema que necesita un alto rendimiento y procesamiento, esta placa puede ser la más adecuada para ese nuevo proyecto.

5.2.1. Arduino MEGA 2560

Arduino es una familia de placas que usan como chip principal los microcontroladores *ATMEL*. Dependiendo del modelo, estas placas ofrecen unas prestaciones diferentes sin salir de unas características comunes. Para facilitar el uso de estas placas para cualquier usuario, incluyen un convertidor *FTDI* para poder comunicar la placa basada en TTL mediante un puerto USB genérico.

La placa *Arduino Mega 2560* usa un microcontrolador *ATMEL atmega2560*, que dispone de 54 *GPIOs (General Purpose Input Output o Entradas y Salidas de Propósito General)*, 16 entradas analógicas con posibilidad de uso para *GPIOs*, un oscilador de cuarzo de 16 megahercios que genera la frecuencia de reloj y 512 bytes de memoria EEPROM independiente a la reservada para el programa. Dispone de 6 *TAUs (Timer Array Unit)*, que para este proyecto se dedicarán a la generación de señales *PWM* para el control de cada uno de los motores. La placa trabaja a una tensión de 5 voltios, pudiéndola alimentar directamente a ese voltaje o a uno mayor a través del regulador de tensión lineal que lleva incorporado.

Uno de los motivos para elegir esta placa ha sido que existe una amplia comunidad de usuarios que la emplean y que han creado niveles de abstracción que facilitan mucho la programación de la placa, sobre todo para gente sin experiencia en la programación de microporcesadores o de lenguajes de bajo nivel. Un ejemplo de esto son las librerías que permiten usar los *timers* para controlar servomotores.

Los *timmers* son contadores que actualizan su valor cada ciclo de reloj aumentando su cuenta en uno. Cuando la cuenta llega a un valore determinado, el *timmer* se reinicia y envía una señal. En este caso, apoyándose en las librerías correspondientes, estos elementos de la placa

Figura 5.7: Arduino Mega 2560.

se pueden usar para general las señales de control de los servos, usando funciones sencillas en lugar de tener que configurar los preescalados de cada *TAU*, la habilitación de los registros correspondientes y la configuración para el valor de desbordamiento del *timer master y slave* que activan el flanco de subida y bajada de la onda *PWM*.

Además, uno de estos *TAUs*, concrétamente el *Timer0*, esta dedicado para funciones que miden el tiempo desde que se arrancá la placa y otras funciones de tiempo que se usan para la gestión del programa. Sobre este contador se basan muchas funciones que permiten controlar algunos flujos del microcontrolador. Es importante señalar esto ya que en el futuro se invocarán a funciones relativas a este *timer* para medir el tiempo de ejecución de algunas rutinas.

Por otro lado, existe una gran cantidad de información sobre estos microcontroladores al ser un proyecto *OpenSource* con millones de usuarios que colaboran activamente alrededor del mundo. Para facilitar las tareas de programación, *Arduino* tiene su propio entorno de desarrollo, que se puede ver en la figura 5.8, que integra la *toolchain* correspondiente para cargar los programas a la placa simplemente pulsando un botón. El lenguaje de programación para estas placas es un lenguaje derivado y simplificado de *C++*, adaptado para estos microcontroladores donde se incluyen la mayor parte de las funciones de este lenguaje, permitiendo, por ejemplo, el uso de objetos.

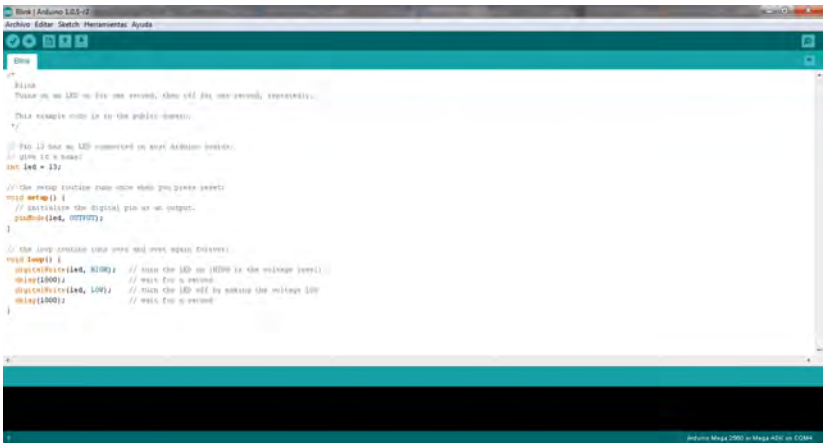


Figura 5.8: Entorno de programación Arduino.

El microcontrolador solo dispone de un núcleo, por lo que solo es capaz de realizar una única tarea al mismo tiempo. Para poder realizar tareas simultaneas o a tiempo real, es necesario

apoyarse en los *timers*, que generarán interrupciones en el programa principal o activarán las salidas de forma automática sin interrumpir el programa, dependiendo de como se configuren previamente por el usuario.

Como cualquier otro microcontrolador, el programa que ejecute esta dividido en dos partes fundamentales. Estas dos partes en *Arduino* son llamadas `setup()` y `loop()`. Dentro de estas dos partes, junto a la cabecera del archivo donde se indica que librerías usar y que objetos globales existen, hay que expresar todo el código. La diferencia entre ambas partes es para que están destinadas en el programa. Mientras que *setup* solo se ejecuta una vez y es considerada la inicialización del programa, `loop` se ejecuta cíclicamente tras el `setup` hasta que se produce un *reset* o se deje de alimentar la placa. En la parte de *loop* se suele ubicar la parte principal del código, ya que el microcontrolador repetirá esta parte de forma continua.

Las 54 *GPIOs* y los 6 *TAUs*son más que suficientes para poder controlar todos los motores y algún otro sensor que pueda ser necesario más adelante.

Shield o placa de expansión

Como se puede ver en la figura 5.2.1, la placa tiene sus conectores agrupados en los bordes de la placa con zócolos o *headers* hembra. Como se ha indicado anteriormente en las tablas 2.1, 2.2 y 2.3, los conectores de los servos son los JR. Estos conectores tienen tres conexiones hembra separados entre sí 0,1 pulgadas o 2,54 milímetros aproximadamente. Para el uso de los servomotores, la mayoría de estos tienen cada una de las 3 conexiones diferenciadas para cada señal. Empezando de un extremo a otro, y siempre en este orden, están las conexiones de *GND, o 0 voltios; Vcc, o alimentación y Señal de control, o signal*. Dependiendo del fabricante, el código de colores cambia, habiendo dos convenios muy extendidos en el mundo del aeromodelismo que se reflejan en la tabla 5.1.

	GND	Vcc	Señal
Convenio 1	Negro	Rojo	Blanco
Convenio 2	Marrón	Rojo	Naranja

Tabla 5.1: Convenio de colores para los conectores de los servomotores.

Hay entender que es necesario un adaptador para poder conectar los conectores hembra de la placa *Arduino* a cada uno de los servomotores. Además, ya que las placas *Arduino* no permiten una salida mayor de 400 mA en el total de sus salidas, es necesario alimentar los motores de forma independiente en lugar de hacerlo a través de la placa. Como posible solución comercial para este problema aparece la placa de expansión *Mega Sensor Shield v2.0*, que se puede ver en la figura 5.9.

Arduino, y los usuarios que lo deseen, diseñan placas de expansión especializadas dependiendo de cada tarea para la que se vaya a usar el microcontrolador. Estas placas de expansión o *shields* se insertan en los zócalos hembra de la placa, replicando la a conexión eléctrica en la nueva placa donde suelen estar alojados los conectores o elementos de la nueva tarea. En el caso de la placa *Mega Sensor Shield v2.0*, tiene la forma exacta de la placa *Arduino Mega 2560*, por lo que basta con colocarla encima para poder usarla.

Esta placa de expansión duplica las conexiones de los *pins* de la placa en la cara superior de esta con tres conectores macho. Cada agrupación de estos tres *pins* corresponde a una entrada o salida de la placa, junto a dos pines de alimentación (`GND` y `Vcc`). Además de estos conectores, la placa presenta dos clemas que permiten alimentar la placa. Para controlar si se quiere alimentar la placa de expansión a través del *Arduino* o usando una fuente de alimentación externa usando las clemas, existe un *jumper* selector. En el caso de que este *jumper* este colocado, cortocircuitará `5V` de la placa con `Vcc` de los tríos de pines.

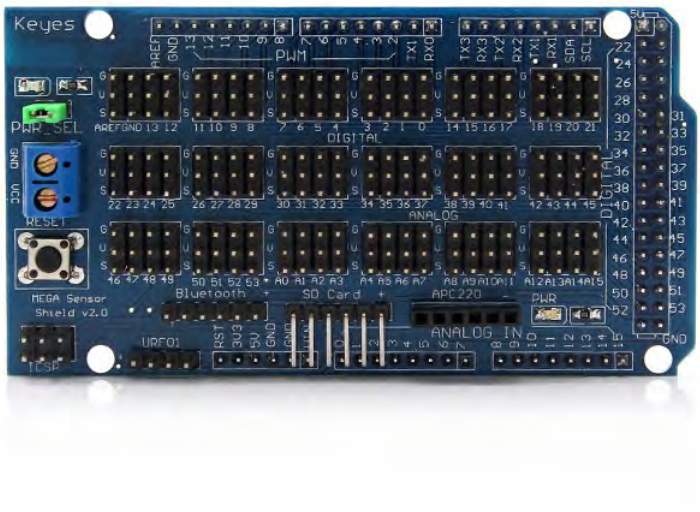


Figura 5.9: Placa de expansión Mega Sensor Shield V2.0.

La disposición de estos elementos facilita la conexión de cada servo con la placa. Para este proyecto, la salida del regulador se conectará a las clemas o directamente a un grupo de conexiones que quede sin utilizarse para alimentar la pista de `Vcc` y asegurarse de que `GND` es común para todo el circuito. Usando cada grupo de tres conectores, se consigue alimentar a través del regulador cada servo, al mismo tiempo que se pueden mandar señales diferenciadas a cada motor, permitiendo gobernarlos independientemente.

Con la idea de tener un robot autónomo, es necesario alimentar la placa *Arduino* dentro del propio robot, y para ello se presentan dos opciones. La primera consiste en hacer una pequeña modificación a la placa de expansión, que consiste en crear una conexión extra soldando un cable entre `Vcc` del *shield* y el conector `Vin` de la placa. Esto permitirá alimentar de una forma segura a la placa a 6 voltios a través del regulador de tensión que incluye la propia placa.

La segunda opción, más indicada para quien no quiera hacer una modificación permanente en la placa o no desee hacer una soldadura, consiste en usar el *jumper* para cortocircuitar a 6 voltios la entrada 5V de la placa. Pese a no ser recomendable aumentar el voltaje al que trabaja el microcontrolador, solo provoca un eleve calentamiento en el chip y acelera mínimamente la velocidad de ejecución de la placa, acortando a largo plazo la vida útil de la placa.

Como contra, este *shield* presenta un fallo grave de diseño. Al introducir por completo la placa de expansión el los zócalos, una de las soldaduras de la *shield* toca con la carcasa del conector *USB* de la placa. Esto provoca un cortocircuito entre `GND` y `Vcc`, pudiendo provocar la inmediata ruptura de cualquier elemento eléctrico conectado al robot. Como solución a este problema, es necesario colocar un aislante entre dichos conductores, como se ve en la figura 5.10. Este aislante deberá separar por completo ambas pistas eléctricas para evitar el cortocircuito de las dos líneas de alimentación. Como consecuencia, la placa de expansión no se podrá insertar completamente en los zócalos.

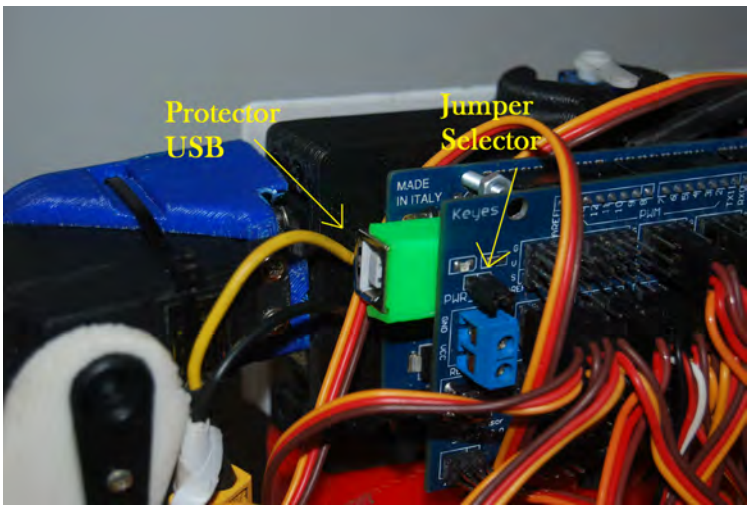


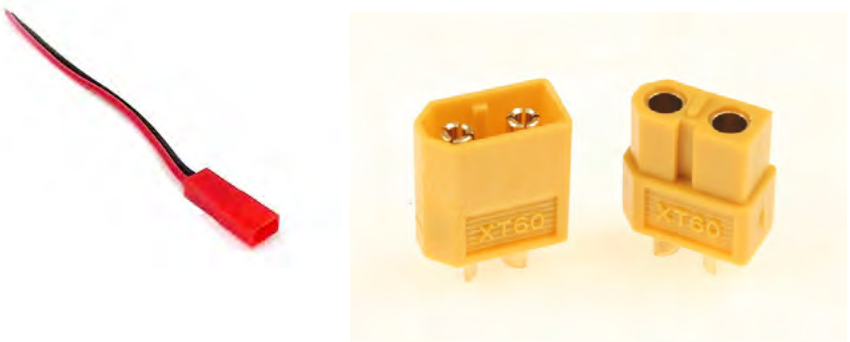
Figura 5.10: Protector USB y jumper selector de alimentación.

5.3. Esquema eléctrico básico y adaptaciones

Con lo explicado en este capítulo, queda definido el esquema eléctrico básico que se va a estudiar en este proyecto. Todos los elementos ya han sido explicados, por lo que en este apartado solo se mencionará las conexiones presentes entre ellos y alguno de los arreglos necesarios para poder conectarlos entre si.

5.3.1. Adaptaciones para la compatibilidad entre elementos

Para poder conectar algunos elementos es necesario hacer leves modificaciones para hacerlos compatibles, como es el caso de la batería y el regulador. Muchas de las baterías Lipo del mercado presentan conectores hembra del tipo *XT 60*(figura 5.11(b)), ya que muchos modelos de aeromodelismo usan este conector. En cambio, el conector JR (5.11(a)) con el que se alimenta el *UBEC Turnigy 8A* es un modelo genérico.



(a) Conector JR. Entrada del regulador.

(b) Conector XT60. Salida de la batería.

Figura 5.11: Diferentes conectores para UBEC y batería.

Para poder conectar estos elementos, es necesario cambiar uno de estos conectores. En este caso particular, se ha optado por cambiar el conector del regulador por un conector *XT60* macho. Para hacer este cambio basta con retirar el conector viejo cortando el cable, preparar los cables donde se van a realizar la soldadura pelando los extremos de estos y aplicando en la punta estaño y, por último, aplicando calor con el soldador para unirlos definitivamente. Es

recomendable usar termorretractil o cualquier tipo de aislante para proteger la nueva soldadura de posibles cortocircuitos.

Como posible modificación se puede retirar el protector plástico de una de las salidas del regulador para dejar los dos conectores al aire. Estos conectores metálicos se pueden usar para la conexión de la clema de la placa de expansión para alimentarla, simplemente introduciéndoles en el orificio correspondiente y apretando el tornillo con un destornillador plano de punta fina para asegurarse que no se muevan. Al ser elementos conductores con voltajes de alimentación, es obligatorio mantener siempre separados a estos elementos para evitar un cortocircuito. Esta última modificación se puede evitar si se conecta directamente a un grupo de tres pines de la placa que no estén en uso, como se ha explicado anteriormente en este capítulo.

5.3.2. Esquema eléctrico

Con los cambios realizados anteriormente, todos los elementos están preparados para poder conectarlos entre sí. El esquema básico se puede dividir entre diferentes subapartados para una mayor comprensión del conjunto. Estos apartados son:

- **Alimentación**. En primer lugar, la batería es la parte fundamental de este circuito, encargándose de dotar al robot de la energía necesaria.

- **Regulador de voltaje**. Ya que la tensión que ofrece la batería no es constante ni tiene el valor adecuado, es necesario cambiar el voltaje para que pueda ser aprovechada por cada elemento.

- **Controladora**. La placa *Arduino Mega* y su placa de expansión son los elementos encargados de manejar todos los actuadores y los posibles sensores que se deseen conectar.

- **Actuadores**. Los elementos controlados y que generan el movimiento son los servomotores. Estos son alimentados gracias a la batería y el regulador a través de las conexiones de la placa de expansión. Al mismo tiempo, estos servomotores son gobernados por la controladora con la señal PWM correspondiente a cada uno de ellos.

- **Otros elementos**. Como ampliación de este proyecto, se pueden reservar recursos para el posible uso de otros elementos, así como la posible integración de sensores.

Diagrama

Como se puede ver en la figura 5.12, así queda planteado el esquema eléctrico básico para poder de dotar de movimiento al robot MYOD. Por último, es necesario señalar que, los servos al tener incluidos el cable de alimentación y señal, es esquema de conexión de *punto a punto*. Esto provoca no poder tener un único cable de alimentación que viaje de motor a motor, en lugar de tener que tirar un cable de cada servo a la placa como es el caso.

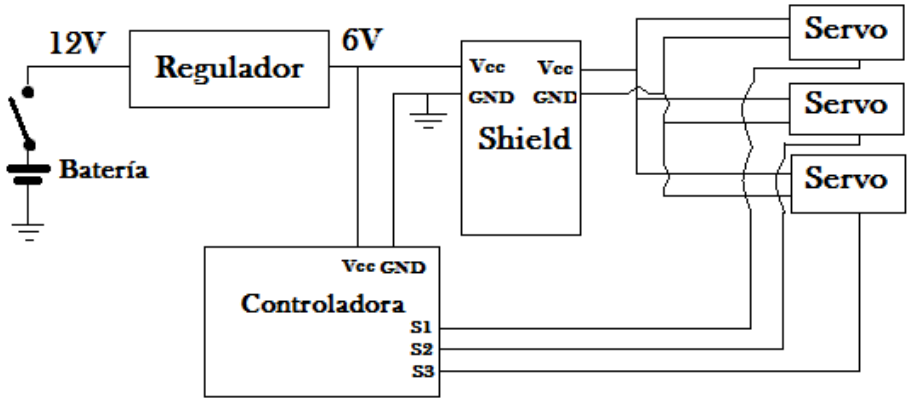


Figura 5.12: Diagrama de bloques del sistema eléctrico.

5.4. Montaje Hardware

Los componentes seleccionados han sido elegidos para poder conectarse entre sí fácilmente. Como se ha dicho, la placa de expansión está diseñada para encajar sobre la controladora, al tener la misma disposición de patillas pero teniendo la primera pins macho y la segunda hembra.

Del mismo modo los servomotores se pueden conectar a la placa de expansión, y por ende a la controladora, introduciendo los conectores en los pines salientes que se encargan de alimentar al motor y de transmitir la señal de control desde la placa.

Por último, el regulador de tensión tiene dos conectores de salida idénticos a los de los servos, salvo que no existe una conexión eléctrica en el terminal que corresponde a la señal. Por eso, se puede conectar también a la placa del mismo modo que un servomotor, facilitando esta conexión. En la entrada de este elemento, se ha colocado un conector *XT60* macho para adaptarlo a la batería, teniendo esta el conector hembra.

Por lo tanto, queda un montaje compacto con pocos elementos diferentes, aunque en gran número. Por este motivo, es necesario agruparlos de alguna manera, con bridas o cuerdas por ejemplo, para que los cables de todos estos no estorben o se enreden.

En la figura 5.13 se puede ver como queda el montaje final de todos los elementos.

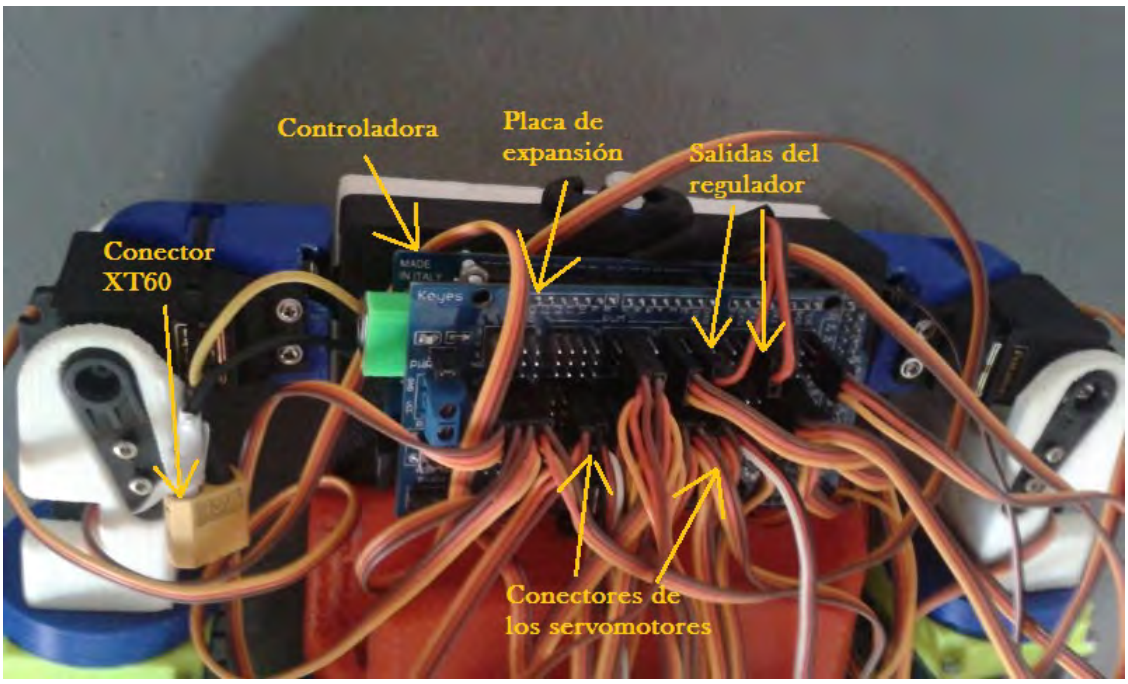


Figura 5.13: Montaje final de la electrónica.

Después de definir el circuito eléctrico del robot y saber que controladora utilizar, se puede empezar a a crear los diferentes elementos del software. La controla seleccionada y los motores del robot imponen las restricciones que hay que tener en cuenta a la hora de crear los programas y sobre que entorno hacerlos.

Capítulo 6

Software propio del proyecto

En este capítulo se hablará de las aplicaciones y librerías que son necesarias para el correcto funcionamiento y configuración del robot. Con estos programas se pueden definir todas la funciones y movimientos básicos del robot para la correcta movilidad de este. Todos los programas y librerías, salvo *Servo.h*, han sido diseñadas a medida del proyecto, aunque se han definido de modo genérico para poder ser utilizadas por otros robot con una configuración o un número diferente de motores. Esto significa que se puede adaptar el mismo código al cambiar unos pocos parámetros para adecuarlo al nuevo problema. Al usar motores gobernados por señal *PWM*, las librerías de control están orientadas a este tipo de señal, sin estar definidas para el control mediante comunicaciones u otro tipo de señales digitales.

6.1. Software de partida

Una de las ventajas que ofrece *Arduino* es que tiene su propio entorno de programación que facilita algunas de las tareas más complejas. En primer lugar, permite utilizar diferentes librerías que facilitan enormemente la programación de ciertas tareas, evitando configurar la placa debiendo modificar registros que controlan periféricos de bajo nivel. Estas librerías permiten a usuarios no familiarizados con la programación a bajo nivel crear programas que satisfagan sus necesidades, apoyándose en funciones que abstraen ese problema.

Por otro lado, ese mismo entorno permite cargar directamente la tarea programada usando un comando. Ante esta facilidad, se puede descartar la idea de usar otros entornos de desarrollo que no este específicamente diseñados para esta placa, al tener que investigar sobre la *toolchain* encargada de todo el proceso de compilación y carga.

En este capítulo también se crearán programas sobre lenguaje `C++` diseñados para usarse sobre un ordenador. Para desarrollar estos programas es recomendable apoyarse sobre algún entorno que facilite la fase de programación, así como de la disposición de alguna herramienta para depurar el código. Existen diferentes compiladores gratuitos, como *DevC++, Qt Creator o Eclipse*, que facilitan esta tarea, recomendando el uso de alguno de ellos para facilitar esta tarea.

6.1.1. Librería Servo

El entorno de programación de *Arduino* incluye una librería, llamada *Servo.h*, que permite generar objetos en el código que son empleados para gobernar servomotores usando señales *PWM*. Estas señales están tratadas para que tenga el período exacto para controlar los motores, además de tener delimitado un ciclo de trabajo máximo y mínimo coincidente con el rango de valores que un motor puede soportar. Para la controladora que se va a usar para el robot MYOD, el número máximo de servomotores que puede controlar es 48, si se dedican todos los *timers* disponibles para esta tarea. Al usar esta librería, pueden aparecer conflictos entre diferentes

partes de un programa, al también emplear *timers* para otras funciones. Por ejemplo, pueden existir incompatibilidades o mal funcionamiento al usar un número elevado de objetos de la clase *Servo* con comunicaciones Serie o I2C, que también utilizan *timers*. Como precaución, los desarrolladores de *Arduino* han intentado diferenciar en sus librerías el uso para el cual están destinados los recursos de los *timers*, evitando así un gran número de posibles incidencias, siempre que no se demande en exceso recursos de este tipo. No obstante, en la página del desarrollador de la librería *Servo.h* se indica que siempre existirán conflictos en ciertos pines para generar señales *PWM*, si no están destinados para el control de servos, dependiendo de el número total de objetos de la clase *Servo* creados.

La librería sólo controla la posición en la que se encuentra el piñón de salida del servomotor, sin poder obtener la posición real donde se encuentra. Para saber en que posición angular se puede encontrar, esta librería devuelve la posición que se está enviando por la salida y que se almacena en un registro. Si se supone que el motor no se encuentra en un punto intermedio de la trayectoria ni está siendo bloqueado mecánicamente por otro elemento, se puede considerar este dato como una buena aproximación de la posición de salida.

Sobre esta librería de *Arduino* se basará la librería que se tendrá que desarrollar para controlar el robot.

Resumen de métodos

A modo de guía rápida, se va a listar los métodos de esta librería de forma resumida y para facilitar su comprensión mostrando de forma simplificada su declaración.

- **Constructor: Servo**. Para crear un objeto de esta clase es suficiente con invocar al constructor y llamar a antojo al nuevo objeto.
- **attach(int pin)**. Vincula el objeto *Servo* pertinente con el pin que se le indique. Existe una sobrecarga de este método de la forma *attach(int pin, int min, int max)*, que permite cambiar localmennte el rango de movimiento de dicho objeto. Como complemento a este método, existe otro llamado *attached()* que devuelve 1 si el *Servo* esta vinculado a un pin o 0 si es al contrario.
- **detach()**. Permite liberar durante la ejecución del programa el pin al cual el objeto *Servo* este vinculado.
- **write(int pos)**. Indica al motor a que posición ha de moverse expresada en grados. Si el valor de posición supera el valor de 540, se considerará igual que la función *writeMicrosecond*.
- **writeMicroseconds(int pos)**. Se usa de modo similar a la función *write*, salvo que hay que expresar la posición en tiempo, en un rango establecido entre los 544 y 2400 μs.
- **read()**. Devuelve el último valor introducido en la función *write* o *writeMicrosecond* grados.
- **readMicroseconds()** Devuelve el último valor introducido en la función *write* o *writeMicrosecond* en escala de tiempo.

6.2. *Trim* o ajuste de posición

Una vez que el robot es montado, las piezas no necesariamente quedan en la posición angular deseada, pudiéndose desviar un par de grados de la posición demandada. Esta falta de exactitud puede ser un problema a la hora de intentar mover el robot, ya que se puede definir una posición angular y alcanzar otra diferente al no haber tenido en cuenta este problema. Una de las causas

es que el piñón tiene un número finito de dientes, no pudiendo colocar cada eslabón en la posición exacta, al tener unas posiciones diferenciadas y definidas por el espacio entre dientes, tanto del piñón como del elemento al que se sujeta.

Afortunadamente, al ser un problema de montaje, se puede prever cuanta deriva se produce al obligar a un motor al ir una posición. Este error es siempre el mismo, a no ser que se vuelva a recolocar la pieza en un futuro montaje, por lo que se puede cuantificar fácilmente y uno se puede adelantar a este problema para corregirlo.

La forma más fácil para paliar este error es dar una posición diferente a la que se quiera alcanzar, sumando a la posición deseada un valor que compense la deriva. Este concepto es el *trim o ajuste de posición*. El *trim* es simplemente un valor numérico, en las mismas unidades que la posición angular, que corrige la falta de exactitud provocada por el montaje. Este valor es diferente para cada elemento que se controle y no es necesario que varíe en el tiempo, salvo que se produzca un fallo en el montaje estructural.

Ejemplo numérico de *trim*

Para aclarar este concepto, se va a mostrar un ejemplo numérico para explicar la necesidad de solucionar este problema con la posición de ajuste.

Por error de montaje, uno de los motores se ha montado desplazado 5º en sentido horario de su posición ideal. Si no se usa el ajuste de posición, cuando se le ordene a dicho motor que se posicione, por ejemplo, a 90º, este se posicionara en $90º + (-5º) = 85º$, siguiendo la fórmula $PosIdeal + Deriva = PosFinal$.

Si se utiliza este ajuste, habiendo cuantificado previamente de forma experimental cual es la deriva, se puede corregir este error. Con los mismos datos que en el párrafo anterior pero esta vez usando el *trim*, el resultado actual es $90º + (-5º) + (5º) = 90º$, siendo la fórmula $PosIdeal + Deriva + trim = PosFinal$. En este caso, la posición a la que se le ordena ir al motor es la que alcanza finalmente, ya que se ha corregido el error de montaje mediante *software*. Para compensar el error es obvio que el tanto el *trim* como la deriva han de tener el mismo módulo y el signo opuesto para poder compensarse.

6.2.1. Implantación

Para este proyecto es vital usar un *trim* para cada articulación móvil, ya que siempre se producen errores de montaje. Con la idea de simplificar el código final y poder usar el mismo archivo de movimientos para diferentes robot del mismo tipo, es necesario diferenciar la posición ideal del *trim*. Al conseguir esto, se pueden usar movimientos idénticos para robots que se hayan montado con diferentes errores de posición. Es importante indicar que este programa se ha basado en la librería *MYOD.h* específicamente diseñada para este proyecto, al tener que usar objetos y métodos que ayudan a la programación. Esta librería será explicada en su totalidad en este mismo capítulo posteriormente.

Al ser algo experimental y propio de cada robot, interesa guardar en un vector estas correcciones para que se puedan usar por el mismo robot en diferentes programas. Para facilitar al usuario la tarea de tener que apuntar a mano el *trim* en cada programa que quiera realizar con el mismo de robot, se ha optado por guardar las correcciones de posición dentro de la propia controladora de forma permanente. Para la realización de esta idea es necesario emplear parte de la memoria *EEPROM* reservada, ya que mantendrá los valores una vez que la placa deje se estar alimentada.

Memoria EEPROM de Arduino

La memoria EEPROM es un tipo de memoria ROM, empleada, por ejemplo, en dispositivos portátiles de almacenamiento, lápices de memoria, etc. *Arduino Mega 2560* tiene reservada

512 bytes de esta memoria, aparte de la necesaria para guardar el programa principal, para cualquier uso que se le quiera dar. Esta memoria viene grabada de fábrica con todos los registros a nivel alto, por lo que es indispensable borrarla para evitar problemas al leer valores que no tenga ningún sentido para el programa.

Dentro del entorno de programación de *Arduino*, es fácil acceder a esta memoria usando programas sencillos. Como se habló anteriormente, los usuarios de *Arduino* han creado niveles de abstracción que facilitan en gran medida la programación de cualquier aplicación. En la misma línea, el entorno de programación de *Arduino* incluye la librería *EEPROM.h* que permite acceder a la memoria *EEPROM* reservada usando apenas tres funciones: una para leer, otra para escribir y otra para poner todos los bit de esa dirección a nivel bajo.

Es importante citar que *Arduino* usa 16 *bit* para representar números enteros de forma genérica. Para la implantación del sistema de *trim* para el robot, se usarán tantos números enteros como servomotores se controlen, en este caso particular 24. Cada número entero ocupa 2 *bytes*, por lo que es necesario usar 48 direcciones de memoria para poder almacenar los datos necesarios. Para guardar estos datos en la memoria, se ha decido dividir el número entero en dos grupos de 8 bits para poder manejarlos adecuadamente, separando el *byte* más significativo del resto. De forma arbitraria, se ha optado por guardar en las primeras 24 posiciones de memoria la parte más significativa del dato, siguiendo un orden concreto a la hora de introducir el ajuste de los motores, y en las 24 siguientes la parte menos significativa de cada valor.

6.2.2. Programa para modificar el *trim*

Una vez que se ha decido como almacenar en los parámetros de ajuste de posición dentro del robot, se puede empezar a escribir el código para un programa que permita gestionar estos valores fácilmente. Para ello, se va a diseñar un programa para *Arduino* que permita variar el *trim* usando únicamente la placa *Arduino* y un ordenador u otro dispositivo para comunicarse con ella. *Arduino* permite generar comunicaciones *UART*, o comunicaciones serie asíncronas, de forma muy sencilla. Aprovechando esta ventaja, podemos usar la ventana de hiperterminal que incluye el entorno de programación de *Arduino* para enviar y recibir comandos e información.

El programa permitirá cambiar uno a uno los valores de la posición de ajuste de cada robot, indicándole para ello qué motor se quiere cambiar y, tras esto, el nuevo valor de ajuste. Para no tener que repetir instrucciones de forma innecesaria, el usuario podrá cambiar repetidas veces el mismo valor de ajuste de un mismo motor hasta que introduzca el comando para seleccionar un nuevo servo. Además, siempre que el usuario lo desee, podrá pedirle al programa que le muestre por pantalla todos los valores de ajuste que estén presentes en el robot en ese momento. Para comunicarse con la placa es necesario abrir la ventana de comunicación serie del entorno de *Arduino* o cualquier otro tipo de visor que permita comunicarse a través de un puerto serie.

Para que se puedan comunicar dos dispositivos con una comunicación asíncrona, es necesario que ambos elementos trabajen a la misma velocidad de comunicación. Por esto, hay que definir en primer lugar cual va a ser la velocidad de comunicación en este programa. Existen numerosas velocidades consensuadas que sirven de *standard* para que sea más sencillo comunicar diferentes dispositivos. En esta aplicación, la velocidad elegida es de 9600 baudios, que es una de las que pertenecen al *standard* y es de las más utilizadas. Además, al ser una interfaz para un usuario, la velocidad de comunicación no necesita ser elevada. Por ser una inicialización, esta declaración de código deberá en la parte de *setup* dentro del programa de *Arduino*.

Tras definir la velocidad de comunicación, el programa realizará la lectura de los valores de *trim* previamente almacenados en la memoria *EEPROM* y los aplicará a cada vez que se imponga una nueva posición. Una vez hecho esto, la placa se conectará con cada servomotor, indicándole con un fichero de comunicación con extensión *.h* en que pin está conectado cada motor y que motor está ubicado en cada articulación. Este archivo tiene gran importancia al

permitir definir de forma clara y sencilla la conexión de la placa con cada pin. Esto además permite tener configuraciones de conexión diferentes para diferentes robots.

Después de la vinculación de los motores a los pines, la controladora envía la orden de que todos los motores vayan a la posición inicial, que se ha elegido 90º al estar en la mitad del abanico de posiciones a las que puede alcanzar cada servo. Para ayudar al usuario, se muestra por pantalla el *trim* actual de cada motor, junto a los dos comandos que permiten controlar el programa. Estos comandos son *C*, que permite seleccionar otro motor para ajustar el *trim* de ese elemento, y *S*, que muestra el valor de ajuste de todos los motores. De forma totalmente arbitraria, el programa al arrancar elegirá por defecto el servomotor del cuello como motor seleccionado. Tras esto, se acaba la inicialización del programa, dando paso a la parte del código que se ejecuta cíclicamente.

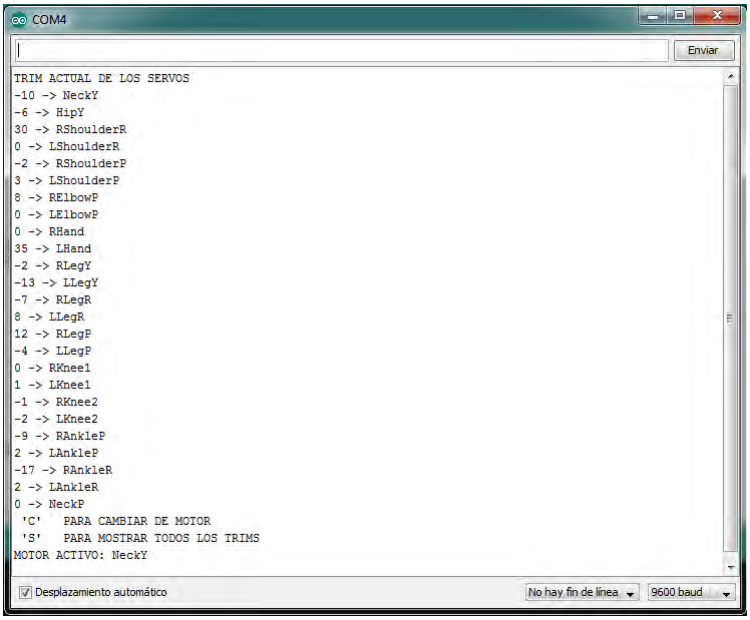


Figura 6.1: Mensaje inicial del programa *trim* a través de la ventana de comunicación de Arduino.

En este momento, el usuario podrá ver por pantalla, como se muestra en 6.1, el ajuste actual de todos los servos. Para modificar el valor de *trim* del elemento que se quiera, basta con teclear el comando correspondiente, concretamente *C*. Tras enviar este comando a la placa, aparecerán listados por pantalla todos los servos que están conectados en ese momento, junto con una etiqueta numérica que permitirá elegirlos, tal y como se ve en la figura 6.2. Para seleccionar el motor al que se le desee cambiar el valor de ajuste sólo es necesario enviar el número correspondiente a dicho motor.

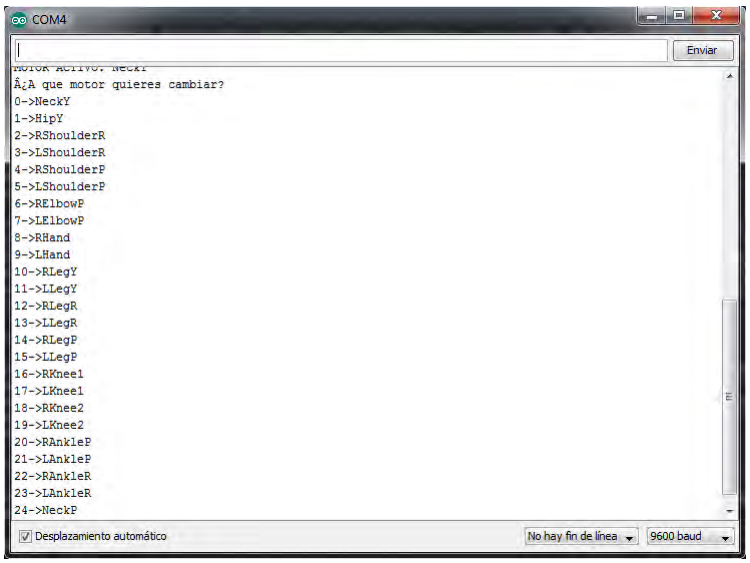


Figura 6.2: Listado de motores que se pueden elegir en *trim*.

Una vez que se haya elegido el motor que se desee cambiar, sólo se necesitará introducir un nuevo valor para modificar el *trim*. Esto se podrá realizar tantas veces como se desee hasta ajustar el valor y decidir seleccionar otro motor, como se ve en la figura 6.3. Cada vez que se introduzca un nuevo valor, la controladora lo guardará de manera automática en las direcciones de memoria correspondientes a dicho motor, para que estén listas para cualquier otro programa que use este sistema.

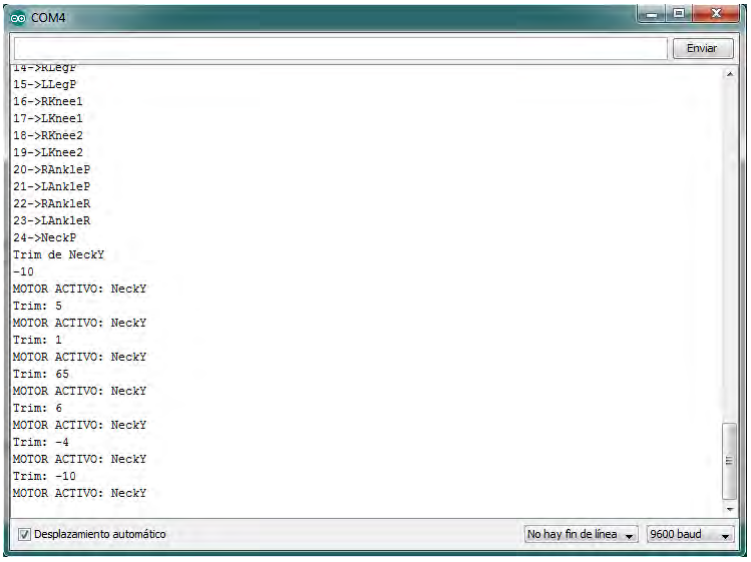


Figura 6.3: Cambios consecutivos del valor de ajuste del motor seleccionado.

Con este programa se consigue ajustar rápidamente y de forma sencilla los valores necesarios para asegurarse que el robot alcanza de forma exacta todas la posiciones a las que se le envíe.

6.3. Librería MYOD en *Arduino* para el control del robot.

Al tener que controlar un número tan elevado de motores a lo largo de la ejecución del programa, es necesario agruparlos de alguna forma para manejarlos de un modo más sencillo. En muchos caso, la velocidad de cada uno de ellos es diferente y han de moverse simultáneamente, por lo que es una complicación añadida al sólo poder procesar una única tarea al mismo tiempo.

En los siguientes apartados se mostrará como solucionar todos los problemas presentes en el control simultaneo de todos los elementos actuadores.

Para poder mover el robot fácilmente es aconsejable crear métodos y funciones que ayuden a manejar el conjunto de servos. Para agrupar estos métodos se ha creado una librería específicamente diseñada para este robot, aunque se puede adaptar fácilmente a otros donde se desee controlar varios servomotores de forma agrupada.

Esta librería llamada *MYOD.h* define la nueva clase *Robot*, que permite controlar al prototipo con los métodos que incluye. Esta librería está basada en *Servo.h*, ya que agrupa varios objetos *Servo* dentro de la clase *Robot*.

Para poder usar esta librería se ha de incluir en el listado de librerías que el entorno *Arduino* puede usar. Para ello, basta con seguir los pasos que se detallan en la página web de *Arduino* [1].

6.3.1. La importancia de la variable tiempo

La necesidad de esta nueva clase está en la importancia de poder controlar todos los servomotores que se deseen de manera simultanea, pudiendo moverlos a velocidades y posiciones diferentes en un mismo período de tiempo. Esto es vital ya que en el movimiento de un robot no todos los motores giran a una velocidad igual ni tampoco alcanzan posiciones iguales. Esto es un reto al tener sólo un *hilo* de procesamiento, ya que no se pueden enviar órdenes simultaneas con una placa *Arduino*. Los motores, al no incluir registros de velocidad, tiempo o posición por no llevar integrado ningún tipo de microcontrolador local para cada uno de ellos, sólo reaccionan ante la señal de entrada en ese mismo instante de tiempo. Esto implica que no se puede grabar en el motor una ruta o trayectoria, por lo que es necesario tener al motor controlado en cada instante de tiempo durante la trayectoria usando la señal de control. Al tener que controlar varios motores, la tarea no es sencilla, ya que se necesita tener actualizada la posición en la que deben estar todos los motores en cada instante de la trayectoria.

Si se usase directamente la clase *Servo*, sólo se podría indicar la posición final que debe alcanzar el motor, desplazándose a la máxima velocidad posible hasta alcanzarla. Esto origina que no se puede controlar la velocidad de movimiento de un motor, al ir siempre a la máxima velocidad permitida al moverse entre dos puntos cualesquiera según ordene el lazo de control del motor. Esto es inviable en la generación de trayectorias, ya que para ello es necesario que los diferentes motores trabajen de forma sincronizada a diferentes velocidades. Por ello es necesario incluir la variable de tiempo en la ecuación, para poder controlar la velocidad de rotación de todos los elementos.

6.3.2. Método *move*

El corazón de esta librería, y posiblemente del proyecto, es el método *move*. Este método permite mover todos los servos a una posición diferente en un período de tiempo seleccionable. Esto ocasiona que la velocidad de movimiento de un servo sea la misma dentro de uno de estos períodos sin importar la posición intermedia en la que se encuentre. Dicho de otro modo, la velocidad angular de cada servo es constante, produciendo un movimiento angular lineal en el tiempo. Además, la velocidad a la que mueve cada servo puede ser igual o diferente, dependiendo únicamente del tiempo total y de las posiciones final e inicial de ese movimiento en concreto. Por ejemplo, a dos servomotores que empiezan en la posición 30º se les ordena ir a las posiciones 40º y 130º en un segundo. El primer motor se desplazara con una velocidad igual a $\frac{40^o - 30^o}{1000ms} = 10\frac{^o}{s}$, mientras que el otro lo hará a $\frac{130^o - 30^o}{1000ms} = 100\frac{^o}{s}$. De modo similar, dos servomotores pueden girar a la misma velocidad partiendo de puntos diferentes.

6.3.3. Principios matemáticos del método *move*

Para conseguir este resultado se ha optado por hacer divisiones muy pequeñas de tiempo en las cuales se actualiza la posición de cada servo. Para saber estas posiciones intermedias sólo es necesario emplear la fórmula 6.1.

$$\alpha_n = \omega \cdot t_n + \alpha_0 \tag{6.1}$$

definiendo α_n como la posición angular en el instante n siendo, a su vez, n un número entero, α_0 la posición inicial del intervalo, t_n un valor de tiempo entre el 0 y T *o período del intervalo.*

Es necesario destacar que n es un número entero que puede toma valores entre 0 y el resultado de 6.2.

$$n_{max} = T/\Delta t \tag{6.2}$$

definiendo Δt como la resolución de tiempo y T como el tiempo total del ciclo o período. En ocasiones esta división puede dar un resultado con decimales, teniendo que redondear siempre al alza. Como inconveniente es que el último intervalo es menor. Por ello, como arreglo, el último valor ha de ser n_{max} con sus decimales para que la posición final sea la correcta.

Por otro lado, ω responde a la ecuación 6.3.

$$\omega = \frac{\alpha_f - \alpha_0}{T} \tag{6.3}$$

quedando representa por α_f la posición final.

Según se incrementa el valor que toma t_n según 6.1, se actualiza de forma creciente la posición de α_n. Si t_n toma el valor 0, se puede observar que corresponde a la posición inicial al quedar

$$\alpha_0 = \omega \cdot 0 + \alpha_0 = \alpha_0 \tag{6.4}$$

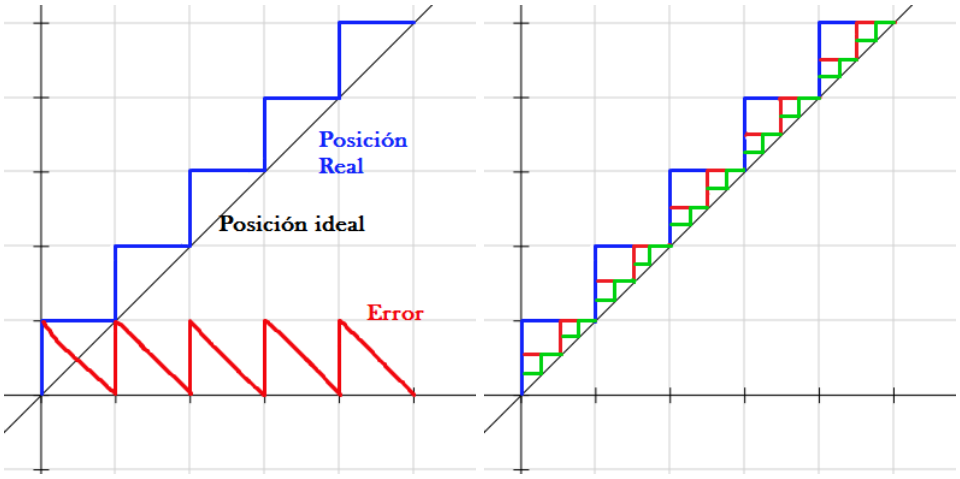
Del mismo modo, si toma el valor máximo correspondiente a T se obtiene la posición final α_f.

$$\alpha_T = \omega \cdot T + \alpha_0 = (\frac{\alpha_f - \alpha_0}{T}) \cdot T + \alpha_0 = \alpha_f - \alpha_0 + \alpha_0 = \alpha_f \tag{6.5}$$

También se puede ver según la fórmula 6.6 que al multiplicar ω por Δt se obtiene el incremento de posición angular para un servo $\Delta\alpha$. Se puede ver que depende de la resolución del tiempo Δt, del tiempo total T, y de las posiciones inicial y final de cada servo, α_0 y α_f. Del mismo modo que las variables de tiempo son iguales para todos los motores, las posiciones dependen exclusivamente de cada motor.

$$\Delta\alpha = \omega\Delta t = (\frac{\alpha_f - \alpha_0}{T})\Delta t \tag{6.6}$$

Al ser intervalos discretos, aumentado en una cantidad fija cada cierto tiempo en lugar de hacerlo de forma continua, no se consigue una trayectoria totalmente recta. La trayectoria resultante se asemeja más a un gráfico de escalera, como se muestra en la figura 6.4(a). Cuanto mayor sea la resolución del movimiento, más pequeños serán los saltos producidos entre posiciones, pudiéndose ajustar más a la recta que se busca. Esta influencia de la resolución se puede ver en la figura 6.4(b). En adicción, al trabajar con números enteros al indicar la posición, su vuelve a producir otro escalonamiento en la respuesta, al no poder enviar cualquier valor por no tener una resolución infinita.



(a) Evolución del error y de la posición. (b) Evolución del error y de la posición.

Figura 6.4: Posición y error de discretización.

6.3.4. Composición de la clase *Robot* en la implementación

Como es lógico, han de emplearse un número de parámetros proporcional al número de servomotores que se deseen controlar. En programación, la forma más fácil de agrupar datos del mismo tipo son los vectores. Estos vectores tienen dimensiones variables en función de como se declaren, pudiendo agrupar tantos elementos del mismo tipo como se deseen. Estos ventores pueden contener variables de cualquier tipo, pudiendo agrupar tanto números enteros sin signo hasta objetos de una nueva clase creada. Para acceder a cada elemento sólo es necesario indicar en que posición se encuentra en el vector. Esto último facilita mucho la programación al poder recorrer un gran número de objetos variando un indexador.

La clase *Robot* esta compuesta por 24 objetos del tipo *Servo* agrupados en forma de vector, otro vector que almacena la posición en la que se encuentran cada servo, de forma similar al método *read* de la clase *Servo* y, por último, un último vector donde se guarda el *trim* de cada servo.

El empleo de otro vector para guardar las posiciones en las que se encuentra cada servo en lugar de usar el método *read* perteneciente a la clase *Servo* es causa del tiempo de ejecución del programa y de la simplicidad del código. El uso de este vector es más sencillo que tener que leer el registro de cada motor, al ahondar menos en los anidamientos de la clase. Esto origina un código más simplificado y reconocible por otros desarrolladores al estar todo definido en la misma librería, a coste de un mayor consumo de espacio de la memoria RAM al almacenar más datos.

El vector de *trim* es también muy importante, ya que se implementa el ajuste de modo sencillo al sumar sus valores en cada movimiento a la posición final deseada. Esta suma es la forma más simplificada en la que se puede implementar el ajuste de posición.

6.3.5. Implementación del método *move*

Para poder usar lo mencionado antes, es necesario pasar estas fórmulas a código de programación, respetando los recursos disponibles. En primer lugar, como se ha explicado antes, es necesario la implementación del *trim* dentro de todas las funciones. Para ello basta con incluir el término de *trim* cada vez que se invoque un nuevo movimiento, sumándolo a la posición final.

También es necesario definir cual va a ser la resolución del tiempo en el que se divide el movimiento. De forma experimental, se ha visto que la función *move* tarda aproximadamente

1,7ms en ejecutarse en *Arduino* al manejar 25 motores. En previsión de que se puedan usar interrupciones que se ejecuten periódicamente para la lectura de posibles sensores que se implanten en un futuro, es necesario dimensionar correctamente este tiempo. Como medida de seguridad, se usará como valor de este incremento de tiempo 5ms. Este valor elegido puede permitir la ejecución del código de una interrupción sin que se produzca un mal funcionamiento, quedando en espera la mayor parte del tiempo de la función *move*.

Tras definir la base de tiempos, se puede implementar fácilmente la fórmula 6.6. Tanto las posiciones como el tiempo total son parámetros variables que son introducidos por el usuario cada vez que se quiera realizar un movimiento lineal. A diferencia de las posiciones angulares que serán el resultado final, este incremento angular interesa que conserve todos los decimales. En el caso de que se perdiesen los decimales por truncamiento, cada posición intermedia no sería la correcta, quedando en una posición menor. Si en vez de ello se redondease al alza, las posiciones intermedias serían mayores a las deseadas. Por ello, para la implementación de este parámetro se usa un variable del tipo número de coma flotante. Como se repite en los demás parámetros, para agruparlos y poder utilizarlos más fácilmente se declararán también en forma de vector.

Para poder definir el avance regular de todos los servos es necesario recurrir a dos divisiones de tiempo. La primera división es la que ocupa el tiempo total del movimiento, y dentro de esta, la segunda formada por las subdivisiones compuestas por la resolución del tiempo, y ambas se pueden ver en la imagen 6.5. Usando el cómputo de tiempo total desde que se arrancó la placa hasta ese momento, el tiempo total y la resolución de tiempo, se pueden definir perfectamente todos los límites para cada ciclo.

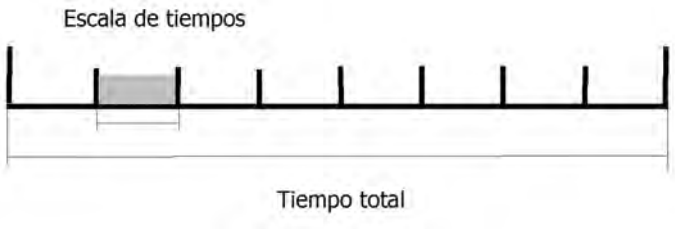


Figura 6.5: Tiempo total y subintervalo de tiempo.

Usando lo anterior, el funcionamiento de este método es el siguiente. En primer lugar, se calcula para cada motor el incremento angular que se debe producir en cada ciclo según la fórmula 6.6. Tras esto, apoyándose en el tiempo transcurrido hasta ese momento, se suma a dicho valor la duración total que ha de tener el movimiento. El resultado de esta suma servirá como el límite que no ha de superarse en la ejecución del método, ajustándose al tiempo indicado. Hasta que no se alcance este límite, se ejecutará cíclicamente las subdivisiones de tiempo. En ellas, usando de manera similar al límite de tiempo total, se calculará el límite de tiempo de ese subciclo en concreto. Para obtener este dato, en cada subdivisión se tendrá que sumar el tiempo actual la base de tiempo. Dentro de cada subciclo, se actualizará la posición para ese instante de todos los motores, al saber en que iteración se encuentra usando para ello un contador. Cada vez que se supere el límite de tiempo de cada subdivisión, el contador se incrementará, actualizando de nuevo las posiciones de cada motor y comprobando se se ha superado también el límite total del movimiento. Por último, se actualizará el valor del vector de posiciones para que coincida con la posición final, pudiéndose utilizar en el futuro.

Como se ha dicho anteriormente en esta misma sección, el tiempo de ejecución de cada subciclo es aproximadamente de 1.7ms para el cálculo del mismo número de motores del robot. Por lo tanto, si está definida una base de tiempo de 5ms, existirá un tiempo en el que no se

realizará ninguna tarea. Durante este tiempo, si no se usan interrupciones, el microcontrolador no ejecutará ninguna otra tarea, comprobando únicamente si se ha sobrepasado o no el límite de cada subdivisión.

6.3.6. Flujograma del método *move*

En la figura 6.6, se muestra gráficamente el funcionamiento de este método de la clase *Robot*.

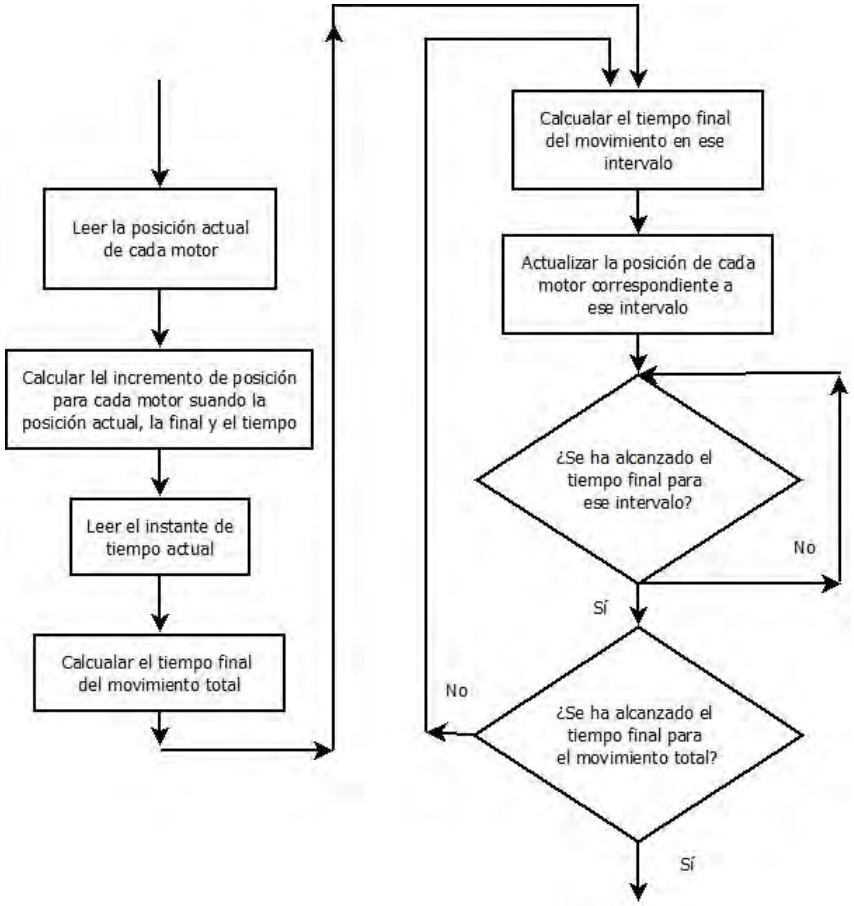


Figura 6.6: Flujograma de la función *move*.

6.3.7. Métodos de la clase Robot

La clase *Robot* no sólo presenta el método *move*, si no que incluye otras funciones propias de la clase para mover de modos diferentes uno o todos los motores de forma independiente o no al tiempo. Además replica ciertas funciones de la clase *Servo* pero haciéndolo para varios motores a la vez.

Para indicar a que pines están vinculados que motores, existe la función *attach*. De modo similar al método de la librería *Servo*, enlaza cada objeto *Servo* de la clase *Robot* con su correspondiente pin de salida, pero esta vez pudiendo unir todos los servos al mismo tiempo.

En contrapartida, existe del mismo modo el método *detach*, que libera los pines usados por todos los motores.

Para poder guardar el *trim* dentro del vector del objeto *Robot*, se emplea el método *trimming*, junto al vector de *trim* correspondiente.

Además de la función *move* que permite mover todos los servos indicando el tiempo de duración del movimiento y la posición de todos los elementos, existe otro método que realiza algo similar. En el caso de la función *moveOffs* en vez de introducir la posición final se introduce

un incremento. Al usar este método, los servos correspondientes se moverán de la posición actual el incremento que les corresponda. Por ejemplo, si un servo se encuentra en la posición de 60º y el valor correspondiente a la función *moveOffs* es -10º, el servomotor se moverá en el tiempo indicado a la posición 50º.

En ocasiones sólo se puede querer mover un único servo dejando los demás en la posición que ocupan. Como esto puede ser de gran utilidad, la clase también incluye métodos para poder realizar estos movimientos. En concreto existen cuatro, que permiten mover un único servo por posición absoluta o relativa pudiendo moverlo a máxima velocidad o en un tiempo controlado.

En el caso de querer mover un único servo a máxima velocidad a la posición a la que se mande, se ha de emplear la función *moveOne*. Si en vez de hacerlo a máxima velocidad se prefiere que llegue en un cierto tiempo a velocidad constante es necesario emplear *moveOneTime*. Si por algún motivo no se desea que un servo se mueva un cierto giro respecto a su posición actual a máxima velocidad, se puede usar el método *moveOffsOne*. Si por el contrario se prefiere que realice la misma tarea a una velocidad constante, *moveOffsTime*.

Existe un último método que permite leer la última posición guardada en una de las posiciones del vector donde se almacenan de un motor en concreto. Este método es llamado *readPos*.

Como precaución, todos los movimientos de la librería están limitados entre 10º y 170º. Esta protección se incluye al usar el *trim*, ya que se pueden mandar a una posición a un servomotor fuera de rango, pudiendo deteriorarse. La función encargada de hacer esto lee tanto la posición a la que se le envía como al *trim* correspondiente. Si en algún caso la suma de ambos queda por encima de 170º, la posición se limitará de forma automática a 170º. Del mismo modo que si desciende por debajo de 10º, se impondrá el límite en ese valor.

6.3.8. Resumen de métodos

A modo de guía rápida, se listan a continuación todos los métodos disponibles en la clase *Robot*.[1]

- **Constructor: Robot**. Para crear un objeto de esta clase es suficiente con invocar al constructor y llamar a antojo al nuevo objeto. Existe una sobrecarga de este método de la forma *Robot(int trim[ ])* que permite crear el objeto con el *trim* incluido.
- **attach(int pin**[ ]). Vincula cada objeto *Servo* con el pin que se le indique.
- **detach()**. Permite liberar durante la ejecución del programa todos los pines que se hayan usado como salidas para los motores.
- **trimming(int trim**[ ]). Permite guardar el *trim* dentro del objeto.
- **readPos(int motor)**. Devuelve el valor de la posición del motor indicado.
- **move(int time, int pos**[ ]). Este método permite ubicar a todos los servos en la posición indicada, necesitando para ello el tiempo que ese elija.
- **moveOffs(int time, int offset**[ ]). Mueve cada la variación indicada en el tiempo que se elija.
- **moveOne(int motor, int pos)**. Mueve a la posición indicada un servo en concreto a máxima velocidad.
- **moveOneTime(int time, int, motor, int pos)**. El servomotor indicado se moverá en un tiempo determinado a la posición que se le pida.

[1] Para diferenciar los elementos singulares de los vectores, se marcará con *[ ]* a estos últimos.

- **moveOffsOne(int motor, int offset)**. El servomotor que se indique se desplazará de su posición actual el valor que se le pida, realizándolo a máxima velocidad.
- **moveOffsTime(int time, int motor, int offset)**. Variará si posición tanto como se le indique en un tiempo determinado.

6.3.9. Generación de trayectorias complejas

Los métodos *move* y *moveOffs* permiten generar trayectorias con una única velocidad angular por motor. A la hora de generar trayectorias reales, los motores han de variar la velocidad angular a la que giran para adaptarse a ella. Como arreglo a esto, se propone linealizar por tramos la trayectoria. Cada uno de estros segmentos ha de presentar una única velocidad para todo el tramo, debiendo generar puntos intermedios para recalcular la velocidad en cada división.

6.4. Generador de caminatas

El *Generador de caminatas* es un programa diseñado sobre el lenguaje *C++* que permite generar un ciclo de una caminata pudiendo parametrizar las variables más significativas.

Dependiendo de que estudio del *movimiento de balanceo* y las diferentes trayectorias que se elijan, las ecuaciones que definen el movimiento cambiarán el resultado final. Por esto, se diferenciarán cada estudio en un programa diferente para evitar tener un código muy complejo. Por lo tanto, este generador no es realmente un solo programa, si no un conjunto de aplicaciones que definen todos los posibles movimientos dependiendo de la trayectoria de cada uno de los subimovientos y el estudio de balanceo que se elija. No obstante, todos los programas son muy similares entre sí al tener una estructura y funcionamiento común, siendo únicamente diferentes para las ecuaciones que definen los giros en las articulaciones.

Estos programas son capaces de generar la trayectoria en caminata correspondiente en función de unas variables introducidas por el usuario. Estas variables definen exactamente todos los parámetros de la caminata, como el periodo y las amplitudes máximas de cada submovimiento. Además, estos programas son capaces de escribir y leer ficheros, por lo que pueden leer configuraciones de los motores para adaptar cada motor a la posición correspondiente de los vectores de salida y, lo más importante, permite guardar las trayectorias en un fichero de texto que se puede leer directamente por el entorno de *Arduino*.

Esto último es una gran ventaja al tener los datos de salida en un fichero sin tener que modificarlos o traducirlos para que sean leídos por la controladora, al respetar el formato que se usa en la librería *MYOD* y, por tanto, en el entorno de *Arduino*.

Para el correcto funcionamiento del programa y tener separados las configuraciones de los archivos generados, es necesario disponer en la misma carpeta donde se ejecute el programa otras dos carpetas, una llamada *config* y otra *exported*, donde se guardarán las configuraciones y los movimientos generados respectivamente.

6.4.1. Funcionamiento

Estos programas tienen partes completamente definidas que se encargan de hacer tareas diferentes. El flujo del programa es siempre lineal, esto quiere decir que no se vuelve al paso anterior en ningún momento, salvo que se introduzcan datos de entrada erróneos e incongruentes. Los siguientes estados de este programa se pueden ver a continuación.

- **Seleccionar/Crear configuración del robot**.
- **Introducción de parámetros**.

- **Cálculo de las trayectorias**.
- **Exportación de los datos**.

Seleccionar/Crear configuración del robot

Esta es la primera parte del programa y es la encargada de reconocer en que posición se encuentra cada articulación dentro del vector de posiciones. Dependiendo de la configuración que se elija, cada articulación puede ocupar una posición diferente, pudiendo generar trayectorias con configuraciones diferentes en el número de motores o la posición que ocupa cada uno de ellos en el vector.

Como se ha visto, el robot que se trata en este proyecto tiene 24 grados de libertad rotativos controlados por servos, y cada uno de ellos está destinado a una articulación concreta. Por ello, es necesario indicar de algún modo que servomotor ocupa cada posición del vector para asignar los valores de movimiento correctamente.

Este punto del programa es el encargado de solventar este problema. Para poder asignar los vectores a sus respectivas posiciones se pueden crear nuevas configuraciones que se ceñan a cada robot. Con esto se puede conseguir controlar robot que tengan un número diferente de grados de libertad al poder decidir que articulaciones están presentes en otro robot que quiera utilizar los mismos cálculos.

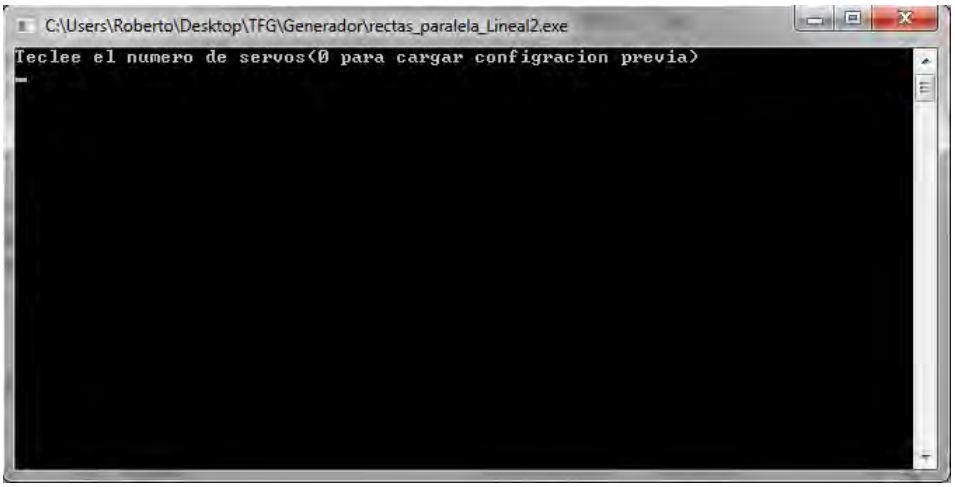


Figura 6.7: Inicio del programa generador.

Al inicio del programa, este pide al usuario que introduzca un número igual a los servos que desea controlar o, en el caso de tener una configuración previa, seleccionar una configuración ya guardada, como se ve en la figura 6.7.

En el caso de optar por introducir un número de servos, el programa mostrará a continuación todas las configuraciones posibles que ofrece el programa para vincular a cada motor con una posición del vector. Para ello, el programa muestra las opciones de las cual dispone junto a una etiqueta numérica. Tras esto, irá pidiendo en orden al usuario que articulación, y por lo tanto que posición del vector, corresponde con cada motor hasta haber introducido el último de ellos.

Una vez hecho esto, el programa pedirá un nombre para guardar la configuración para usarla en un futuro. Esta configuración se guardará como un archivo de texto plano donde aparece por orden los datos introducidos. La forma de guardar los datos corresponde a incluir la etiqueta numérica seguida del carácter ';'. Este carácter permite diferenciar las diferentes etiquetas al separarlas con un elemento no numérico fácil de identificar por el programa. Estos archivos de configuración se almacenan en la carpeta *config* y tienen la terminación *.conf* . Estos archivos son guardados con el mismo nombre que el que ha sido introducido por el usuario y es guardado

en formato de texto plano para, en caso de ser necesario, poder modificarlo con un editor de texto en lugar de tener que generar una nuevo.

Además de guardar la configuración, el programa la tomará como referencia para el resto de la ejecución del programa y mostrará la configuración por pantalla.

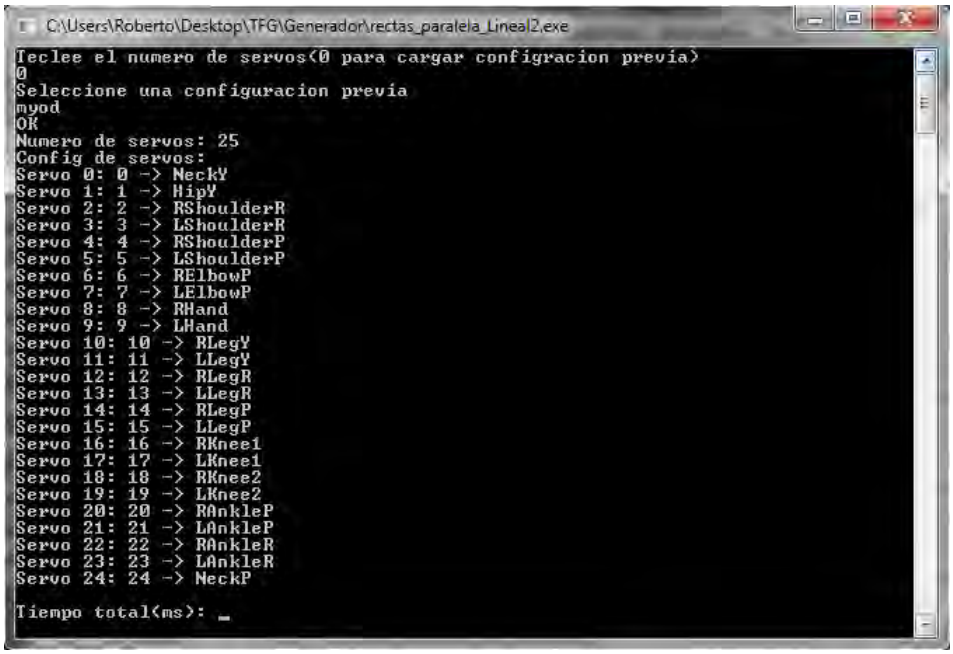


Figura 6.8: Listado de motores de una configuración.

En cambio, si el usuario introduce el número 0 cuando el programa pide un número de servos para una nueva configuración de motores, el programa entrará en la opción de cargar una configuración previa, como se ve en la figura 6.8. Por esto, el programa pedirá al usuario que introduzca el nombre de una configuración que este guardada previamente en la carpeta *config*. Si el usuario introduce un nombre que coincida con un archivo que este guardado en dicha carpeta, el programa lo leerá y cargará la configuración para usarla durante la ejecución del programa. En el caso de no encontrar ningún archivo que coincida, el programa indicará esto mostrando un mensaje de texto por pantalla indicando esto y volviendo a mostrar el mensaje de inicio del programa al reiniciarse el mismo.

Introducción de parámetros

Este estado del programa es el que pide al usuario que introduzca los parámetros que definen la trayectoria a seguir. Estos parámetros, como se ha visto a lo largo del capítulo 4.2, son las amplitudes máximas de cada submovimiento así como el tiempo total del movimiento y la división del mismo en intervalos más pequeños con la misma duración.

En primer lugar, el programa pedirá al usuario que introduzca el tiempo total que se quiere que dure el periodo de la caminata. Tras esto pedirá el número de intervalos en los que se quiere trocear el tiempo total para conseguir generar más o menos pendientes que se amolden mejor a la trayectoria a seguir. Después de introducir estos dos parámetros, el programa dividirá el primero de estos valores entre el segundo para conocer el tiempo total de cada subdivisión. Este tiempo interesa que al menos sea varias veces superior a la base de tiempos que se utilice en la librería *MYOD*. Si el tiempo de los intervalos es menor, la librería no esta preparada para soportar ese movimiento, pudiendo llegar a dañar algún elemento.

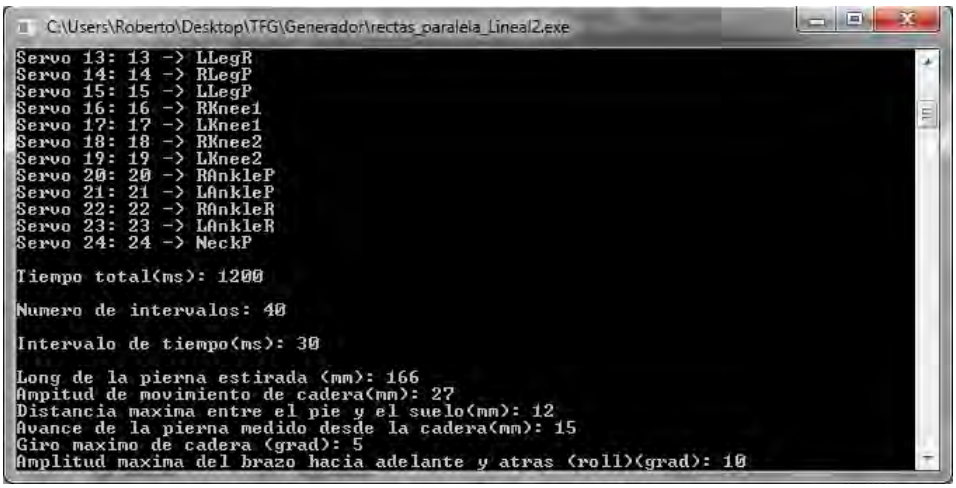


Figura 6.9: Introducción de los parámetros de la caminata.

Una vez que se han introducido los parámetros temporales, es hora de indicar al programa cuales son los parámetros geométricos que definen cada submovimiento, del modo que se ve en la figura 6.9. En este punto sólo es necesario incluir las amplitudes máximas de cada movimiento, ya que, para facilitar al usuario la tarea de introducir y memorizar más datos, los valores umbral se incluyen como constantes en en el programa. Si se deseasen cambiar estos valores limitantes sería necesario volver a compilar el programa tras modificar estos valores. Esto que parece una desventaja realmente no lo es, ya que, como se ha dicho, estos programas están dirigidos al público en general sin que este puede estar familiarizado con este método. Esto significa que, una vez realizados los experimentos pertinentes para hallar estos valores se pueden dejar como una parte no variable del programa, aunque se da la posibilidad de cambiar estos valores al dejar publicados también los archivos fuente. Además de estos parámetros umbral, también se declaran dentro del código cada uno de los desfases de cada trayectoria.

El programa en este punto pedirá al usuario que introduzca los parámetros geométricos que definen el movimiento y al robot. Estos valores hacen referencia a la longitud de la pierna en posición estirada y a las amplitudes de cada uno de los movimientos. El programa siempre pedirá en orden los siguientes parámetros que se listan a continuación.

- Longitud de la pierna estirada (milímetros)
- Amplitud del movimiento de cadera (milímetros o grados[2])
- Distancia máxima entre el pie y el suelo (milímetros)
- Avance de la pierna en horizontal (milímetros)
- Giro máximo de la cadera (grados)
- Amplitud del giro máximo de los brazos (grados)

Cálculo de la trayectoria

Tras introducir todos los parámetros significativos, el programa pasará al siguiente estado, que es el encargado de generar la trayectoria. En este punto el usuario no puede interaccionar con el programa, al ser un estado de cálculo puro.

Para generar las trayectorias es necesario saber tres elementos importantes: qué intervalo hay que calcular, qué articulación es la que hay que calcular y cuáles son las trayectorias y parámetros correspondientes para esa trayectoria.

[2]Dependiendo de la versión y el estudio elegido este parámetro cambia las unidades en las que está referido

Este programa no calcula los movimientos en función directa del tiempo, si no en función de la posición relativa del intervalo respecto al número total de intervalos declarados. Se puede obtener una relación directa entre los intervalos y el dominio de la función sinusoidal de la siguiente forma.

$$\frac{n_i}{n_T} = \frac{\omega t_i}{2\pi}$$

Esto fórmula elimina de la ecuación el tiempo total, generalizando el cálculo sin tener en cuenta el periodo indicado por el usuario o calcular la velocidad angular ω y multiplicarla en función del tiempo. Como se ha dicho, ya se ha calculado el tiempo entre intervalos, por lo que luego se puede volver a trasladar la fórmula en función del tiempo.

La forma de resolver las posiciones de cada articulación por el programa consiste en ir haciendo barridos a lo largo de los intervalos, y dentro de estos un segundo barrido que lee el vector de configuración elemento a elemento para saber cual es la articulación activa. Además de esto, dentro de cada etiqueta de la articulación se encuentran los diferentes estudios de superposición relativos a cada una de ellas. En el caso de tener una articulación que esté presente en varios movimientos, existirán múltiples cálculos que realizar dentro de ese apartado, sumando los incrementos producidos en cada uno de ellos a la posición inicial.

Además, dentro de cada uno de los movimientos de cada articulación se recalcula la trayectoria que corresponde para ese intervalo en función del desfase declarado para esa trayectoria y el movimiento. Una vez que se ha calculado la trayectoria, se calcula el desplazamiento relativo para ese punto dependiendo de la trayectoria recién calculada y multiplicándose por la amplitud de ese movimiento, obteniendo el desplazamiento real para ese instante del tiempo. En el caso de existir otro movimiento presente en esa articulación, se volverá a recalcular una nueva trayectoria para ese nuevo movimiento para volver a realizar el cálculo correspondiente para ese movimiento.

Tras calcular para todos los intervalos la posición de cada articulación según los movimientos, trayectorias y parámetros, el programa mostrará por pantalla cada uno de los vectores calculados para poder verificar los resultados antes de guardarlos en un fichero.

Exportación de datos

Este estado del programa es el encargado de guardar los datos calculados en función de los parámetros de entrada y la configuración del robot y dejarlos almacenados de forma permanente en un archivo de texto individualizado. Este archivo de texto plano tiene una extensión *.h* , para poder ser leído fácilmente por el entorno *Arduino* al estar basado en el lenguaje *C++*.

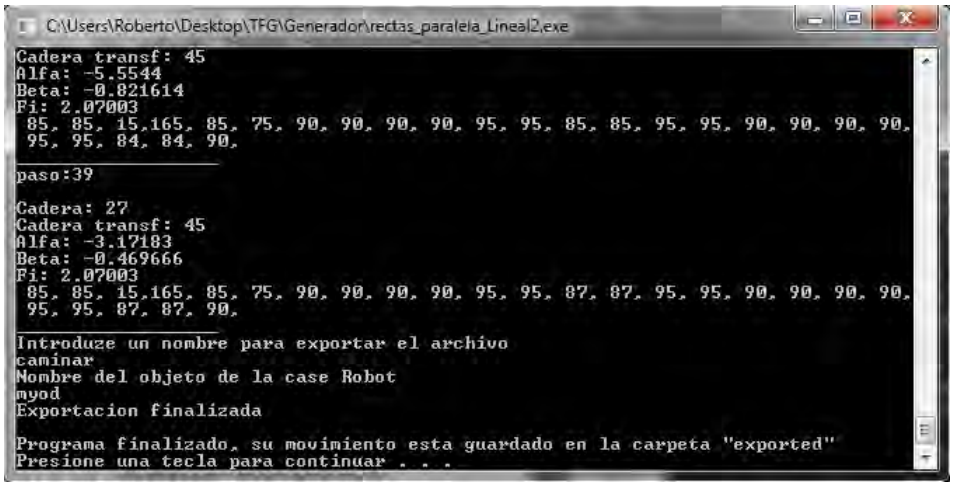


Figura 6.10: Exportación de los cálculos.

El usuario puede dar nombre de la forma que prefiera a este archivo de salida, dándole, de forma automática, el mismo nombre a la función que agrupa a esos movimientos, según lo visto en la figura 6.10. Además, para facilitar la compatibilidad con el entorno *Arduino*, el programa pedirá al usuario el nombre del objeto de la clase *Robot*, declarado en la librería *MYOD.h*, que se utilice en el programa para mover el robot. Además, el programa añadirá tras el nombre del fichero la terminación *_exp.h* . El formato del fichero será detallado más adelante en el apartado 6.4.4.

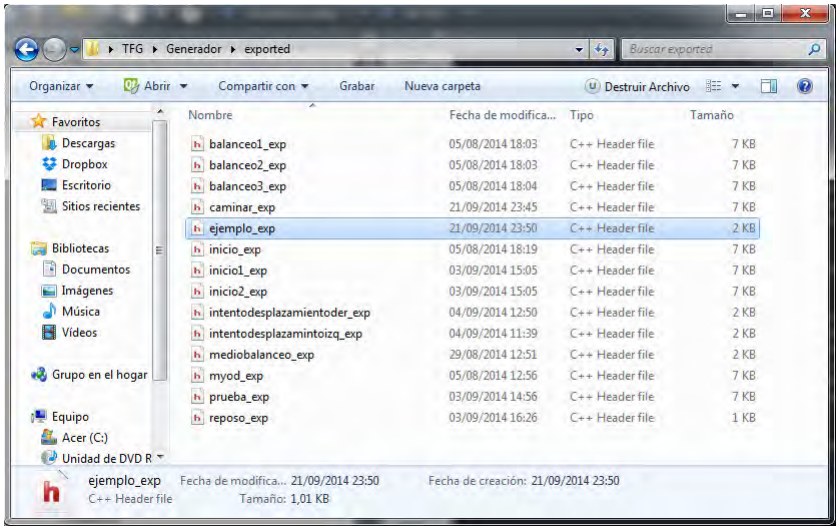


Figura 6.11: Archivos guardados en la carpeta exported.

Una vez que se han dado estos dos parámetros al programa, procederá a guardar los datos en el fichero correspondiente ubicándolos en la carpeta *exported* ya mencionada, como se ve en la figura 6.11.

6.4.2. Flujograma del programa

En este apartado se intenta mostrar gráficamente los estados por los que atraviesa el generador de caminatas para facilitar al lector el funcionamiento de los programas, mostrado en la imagen 6.12.

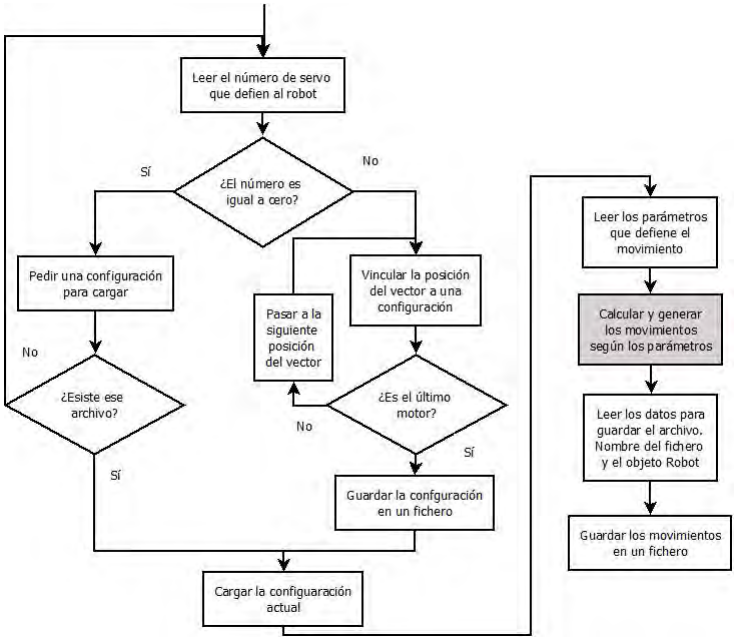


Figura 6.12: Flujograma del programa generador.

6.4.3. Flujograma de la función del cálculo

En la figura 6.13 aparece el flujograma que representa de forma esquemática el funcionamiento de la función encargada de generar las trayectorias. En dicha figura se representan los distintos barridos que hay que hacer para obtener la trayectoria completa, dependiendo del intervalo en el que el programa se encuentre y de cada uno de las articulaciones.

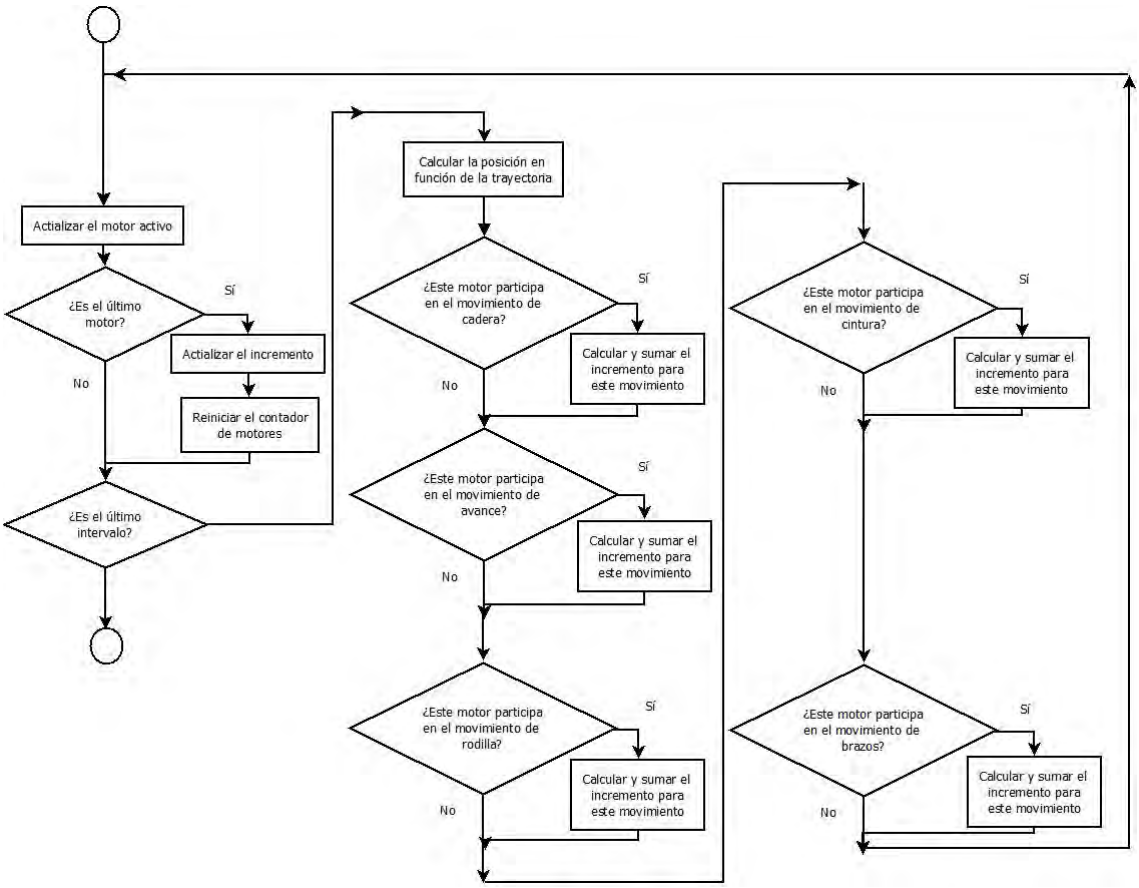


Figura 6.13: Flujograma de la función generadora.

6.4.4. Formato de salida del fichero

El generador de caminatas crea de forma automática ficheros de texto que se pueden usar directamente sobre cualquier programa en el que se quiera usar estos movimientos. Esta compatibilidad se debe a que utiliza los mismos métodos que los explicados en la librería *MYOD*, que a su vez está expresamente diseñada para usarse sobre el entorno *Arduino*. Realmente este fichero guarda la información dentro de una función para *C++*. Dentro de esta función son guardados todos los vectores de posición de cada articulación en todos los intervalos, que posteriormente permiten posicionar al robot de la forma conveniente.

Esto permite usar de manera muy simple los movimientos generados, sólo con incluirlos en la misma carpeta que el archivo de *Arduino* e incluirlo en la declaración del programa.

Los archivos generados están diferenciados en distintas partes, para estructurar la información de una manera clara y sencilla.

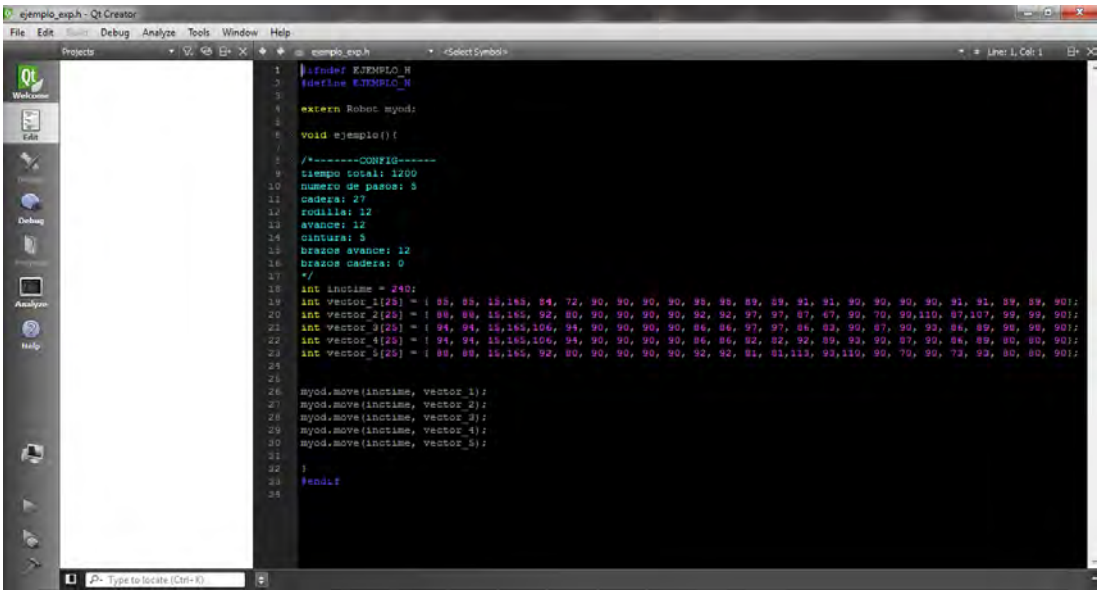


Figura 6.14: Formato de salida del archivo generado.

Como se puede ver en la imagen 6.14, el programa tiene unas partes claramente diferenciadas entre sí, siguiendo una estructura similar a la declaración de un fichero *.h* para incluir parte de un código en *C++*.

Lo primero en aparece en el archivo son dos condicionales para ayudar al programa a compilar correctamente, protegiendo de una posible declaración múltiple de una misma función o elemento del código. Estas dos líneas de código vienen complementadas con la última línea que aparece en el archivo, cerrando la definición condicional.

Tras estas líneas, aparece declarado el objeto de la clase *Robot* al que queremos referir el movimiento. En esta declaración se puede ver que el objeto está declarado de forma externa, al suponer que ya está definido en el archivo principal al considerarlo como un objeto global si la declaración se ha hecho correctamente. Esta declaración de tipo *extern Robot* es importante ya que evita que el compilador no de error al no encontrar la declaración del objeto en ese mismo archivo, permitiendo declararlo en otra parte del código.

Después de esta declaración, se encuentra la función que agrupa todos los movimientos necesarios para esa caminata, según los parámetros de entrada y la configuración del robot. Esta función es del tipo *void* y está declarada según las reglas del lenguaje *C++*. Esta función tiene el mismo nombre con la que se haya querido declarar en el programa de generación y no admite que se le pase ningún valor.

A su vez, esta función esta dividida en tres partes perfectamente reconocibles. En la primera de ella se puede ver dentro de un comentario los parámetros que han definido el movimientos, que han sido explicados previamente en 6.4. Estas líneas de código no afectan en absoluto al programa, pero sirven de referencia al usuario de como ha sido definido el movimiento según los parámetros de entrada.

Tras estas líneas que sirven como recordatorio dentro del fichero, se declara todos los movimientos en vectores que siguen la misma configuración que ha sido cargada por el programa. Estos vectores se enumeran consecutivamente correspondiendo con el lugar del intervalo al que se refieren. Por lo tanto, aparecerán tantos vectores como se haya declarado en el generador, pudiendo tener esta parte del código una extensión variable y, por lo tanto, consumiendo una cantidad diferente de recursos dentro de la controladora. Además, justo antes de estas declaraciones de los vectores de movimiento, aparece la declaración del parámetro que define el tiempo de todos los intervalos dentro de este movimiento, teniendo este parámetro por nombre *inctime*.

Por último, aparecen las ejecuciones de los movimientos. Estos movimientos están definidos sobre la función *move*, que ya ha sido explicada en el apartado 6.3.2 y el objeto de la clase *Robot* que antes se ha declarado. El método *move* está divido en dos partes: la primera define mediante un entero la duración del movimiento y la segunda el vector de posiciones al que se quiere alcanzar. Como se ha dicho anteriormente, el incremento de tiempo entre un movimiento y el siguiente interesa que sean iguales para homogeneizar los movimientos y evitar tener en cuenta una nueva variable. Por eso, el primer parámetro de todos los métodos *move* será siempre la variable *inctime*. En el segundo parámetro de este método están colocados de forma ascendente todos los vectores que antes se han declarado, que corresponden con los movimientos de la trayectoria.

6.4.5. Ejemplo. Incluir el archivo exportado en un programa para *Arduino*

Como se ha dicho repetidas veces, el formato de salida de estos programas es compatible con el entorno de *Arduino*. En este apartado se explicará como usar correctamente estos archivos generados para poder ser aprovechados y generar una trayectoria de caminata para el robot.

En primer lugar, los programas de *Arduino* necesitan estar ubicados en un carpeta la cual tenga el mismo nombre que el propio programa. Si se desean usar archivos que tengan relevancia para el programa por incluir código o cualquier otro tipo de información, se han de ubicar en esa misma carpeta para que el entorno tenga acceso a ella. Este es el caso, el fichero de salido del generador de trayectorias responde a esto. Por lo tanto, cada vez que se quiera usar un archivo de este tipo es necesario ubicarlo en esta carpeta.

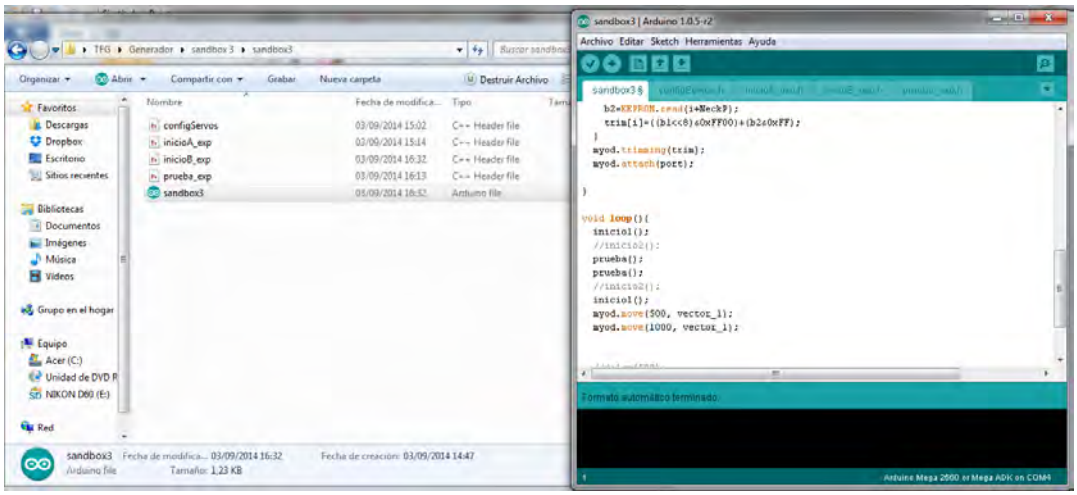


Figura 6.15: Uso real del movimiento exportado y carpeta del programa Arduino.

Una vez colocados los ficheros de ese directorio es cuando se puede abrir el archivo principal de *Arduino* usando su propio entorno. En él se pueden ver justo encima del área de trabajo diferentes pestañas que hacen referencia a todos los archivos que se encuentran es esa carpeta.

El siguiente paso para poder utilizar las trayectorias consiste en llamar a las librerías correspondientes y a los archivos que almacenen movimientos usando el comando `#include`. Si se ha seguido los pasos de que se detallan en la web de *Arduino*, para llamar a la librería *MYOD.h* es necesario usar los operadores '< >', por no estar en la misma carpeta que el archivo principal. Para los demás archivos que se encuentran en esa capeta es necesario usar los operadores ' " "'. En este punto es también necesario cargar la configuración de pines que se ha usado para el ajuste de posición de cada articulación del robot.[3]

Para poder utilizar las funciones que incluyen los archivos de caminata, es necesario tener declarado un objeto global de la clase *Robot* con el mismo nombre que los archivos para poder moverlo correctamente. Además de esto, es necesario cargar el *trim* almacenado en la memoria de la controladora para poder posicionar al robot adecuadamente.[4] Para incluir las posiciones de ajuste y los movimientos es necesario incluir las ordenes que aparecen en la figura 6.16.

Figura 6.16: Procedimiento sencillo para utilizar los movimientos.

Una vez hecho todo esto, basta con llamar a la función guardada en el fichero para poder mover el robot según sus instrucciones. Un ejemplo de esto aparece en la figura 6.15.

6.5. Generador de movimientos por cinemática directa

En ocasiones puede ser necesario generar movimientos para realizar otras tareas con el robot. Estos movimientos pueden ser utilizados para desplazar al robot por el espacio, generando movimientos laterales, giros o subir pequeñas escaleras. Además también se pueden necesitar otros movimientos que sirvan sólo para mover al robot a gusto del usuario. Para facilitar esta tarea surge el *Generador de movimientos por cinemática directa.*

[3] Ver el apartado 6.2
[4] Ver el apartado 6.2

Con este programa se puede mover al robot de una forma sencilla al indicar casi a tiempo real que posición ocupa cada articulación, en lugar de tener que compilar un nuevo código para visualizar el movimiento. Utilizando el *Generador de movimientos por cinemática directa*, se puede mover el robot utilizando diferentes movimientos que se definen el usuario durante la ejecución del programa.

Este programa se ha decidido implantar directamente sobre *Arduino*, aprovechando la terminal de comunicaciones que permite transmitir información de la placa al ordenador o viceversa. Además al utilizar la misma librería y métodos para generar los movimientos, el resultado final del movimiento es el mismo que el que se ve mientras se manipula el robot. Si se respeta el archivo de configuración de pines y en *trim* almacenado en la controladora se sigue el mismo procedimiento para usar el archivo que cuando se usa un archivo generado por el *Generador de caminatas*. El inconveniente principal que surge al utilizar esta terminal es que no se pueden guardar ficheros de forma automática en el ordenador con la placa *Arduino Mega 2560*, aunque este problema se solucionará posteriormente con otro programa diferente. Otro inconveniente de usar esta plataforma es que no te permite añadir movimientos de forma dinámica, al no tener el microcontrolador esta opción disponible, por lo que se han de determinar previamente por el usuario el número de movimientos cada vez que se ejecute el programa.

6.5.1. Funcionamiento

Este programa guarda una gran similitud con el programa de *trim*, tanto en la forma de manejarse como en la estructura interna y funcionamiento. La diferencia entre uno y otro es que este no graba los datos de ajuste, si no que gestiona una matriz de datos donde se guardan las posiciones de cada uno de los movimientos apoyándose en los valores de *trim* ya guardados para poder utilizar el mismo movimiento con otro robot.

De hecho, el primer paso en este programa y en el de *trim* es idéntico, al leer los valores de ajuste de posición y el archivo que relaciona los servos con los pines de la placa, inicializando así el robot.

Tras esto, el robot pedirá al usuario por la terminal de comunicación el número de movimientos que desea tener el programa, como se ve en la figura 6.17. Si el usuario teclea un valor incorrecto o un número negativo, el programa volverá a pedir un nuevo valor hasta que este sea un número entero positivo. Tras introducir este valor, se creará de forma automática una matriz con tantas columnas como motores más uno, siendo este añadido el valor de la duración del movimiento, y tantas filas como movimientos. Esta matriz será inicializada por defecto con todos sus elementos a 90, incluido el tiempo.

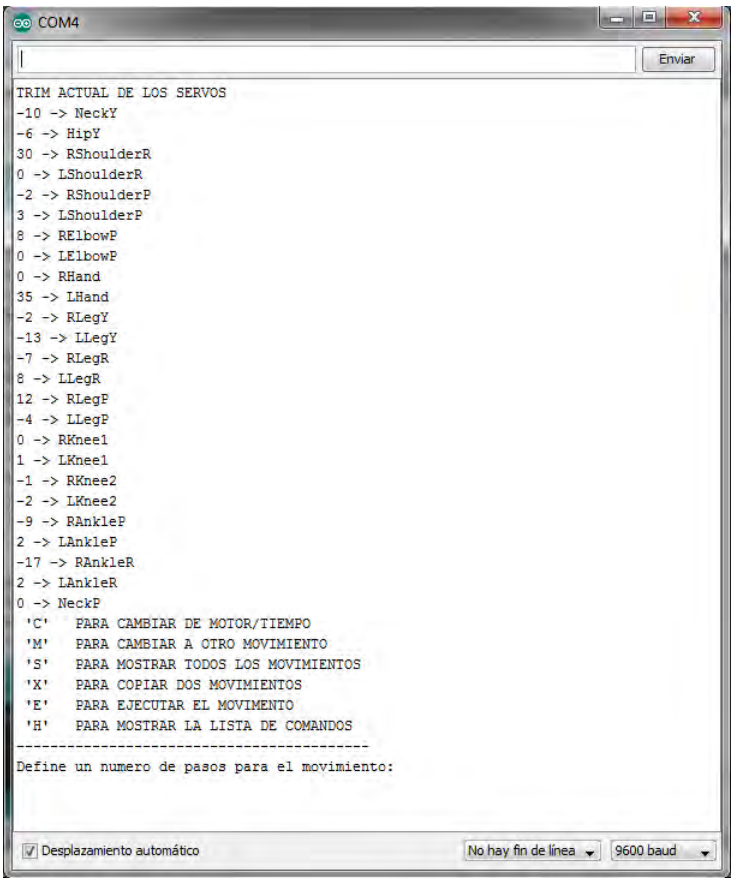


Figura 6.17: Mensaje inicial del programa *Generador de movimientos por cinemática directa*, donde se pide al usurario el número de movimientos que se desean programar.

Una vez inicializado el robot y el tamaño de la matriz, es cuando se pueden definir las posiciones y valores del tiempo en cada movimiento. Para ello existen diferentes funciones que se muestran a continuación.

- **Introducir un valor**
- **Cambiar de motor/tiempo**
- **Cambiar de movimiento**
- **Copiar un movimiento a otro**
- **Ejecutar los movimientos**
- **Mostrar todos los movimientos**
- **Ayuda/Mostrar todos comandos**

Introducir un valor

Si se teclea un número fuera de estas funciones, el programa lo reconocerá como un cambio de valor en el elemento de la matriz correspondiente al movimiento y motor seleccionado. Por esto, si se introducen varios valores de forma consecutiva, se cambiará el mismo elemento, como se ve en la figura 6.18. Además de cambiar el valor numérico de la matriz, el robot leerá ese valor y mandará a la articulación correspondiente a dicha posición. Esto permitirá visualizar al usuario la posición de ese elemento a tiempo real.

El programa trabaja internamente con cadenas de caracteres de tres elementos, por lo que el número máximo que se puede introducir en este elemento no puede ser mayor que 999. En

el caso de los motores no representan ningún problema, ya que la librería que controla a los servos limita el rango a [10,170]. Pero no es así en el caso de la variable del tiempo en cada movimiento, que sólo está limitada en los números negativos. Por la característica de la cadena de caracteres de tres elementos, el valor máximo que puede almacenar es de `999`. Posteriormente se mostrará como variar este límite fuera de este programa.

Figura 6.18: Cambio de valor del elemento seleccionado al introducir un valor.

Cambiar de motor/tiempo

Esta función del programa es la encargada de seleccionar los elementos de de la fila activa de la matriz. Esto quiere decir que es la encargada de ir seleccionando los motores dentro de un mismo movimiento. Esta función es casi idéntica a la del primer programa aquí mostrado al poder elegir el valor de posición de cada uno de los motores, pudiendo recorrer cada uno de ellos al antojo del usuario.

Para ello, es necesario enviar el comando `C`. Tras recibirlo, el robot mostrará por la terminal el tiempo y todos los motores seguidos de una etiqueta numérica, según lo que se puede ver en la figura 6.19. Esta etiqueta permite seleccionar cada uno de ellos introduciendo el valor correspondiente a continuación. Si el número que recibe el robot está dentro de la lista, pasará a seleccionar ese elemento activo que será el que se pueda modificar. En el caso de que el número no esté en la lista mostrada, el programa volverá a seleccionar como activo el motor que antes lo estuviera. Es importante destacar que el tiempo también forma parte de este listado, por lo que también se puede seleccionar este elemento del vector para modificarlo posteriormente.

De forma predeterminada, el elemento activo por defecto en la primera ejecución del programa será el tiempo del primer movimiento.

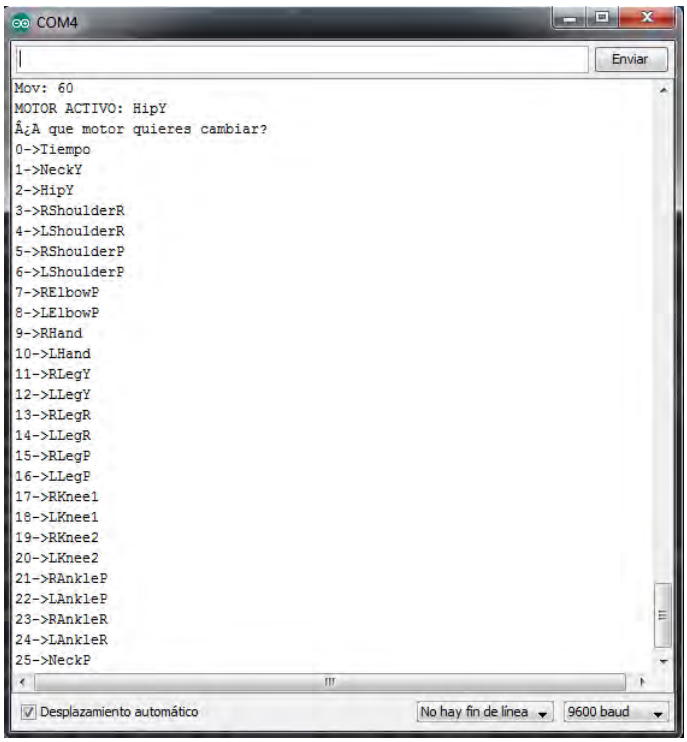


Figura 6.19: Listado de motores que se pueden seleccionar, incluyendo el tiempo.

Cambiar de movimiento

Con esta función se puede seleccionar un movimiento para poder modificar los valores de posición que este incluye. Esto permite recorrer las filas de la matriz donde se almacenan las variables, pudiéndolas seleccionar una a una para luego modificarlas con el método anterior.

Para usar este función es necesario enviar el comando `M`. El robot enviará como respuesta un listado de todos los movimientos que existen en la matriz, mostrando el valor de todos los elementos, tal y como se ve en la figura 6.20. Junto a esta información aparece una etiqueta numérica para poder diferenciar un movimiento de otro. Después el usuario deberá introducir el valor correspondiente de uno de los movimientos para acceder a él. Si no es así, el programa volverá a tomar como movimiento activo el anterior.

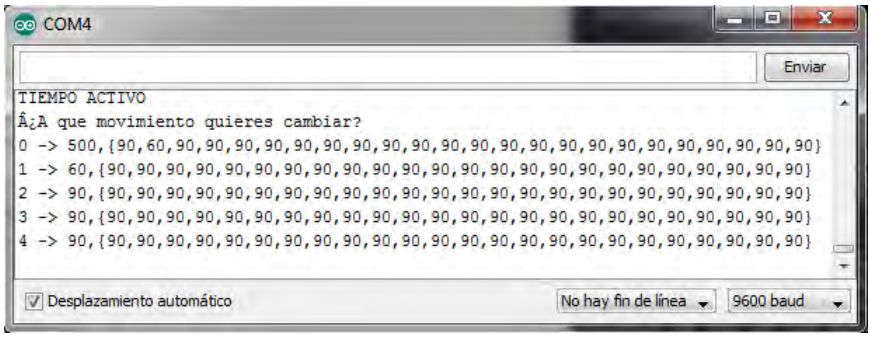


Figura 6.20: Listado de movimientos disponibles.

Copiar un movimiento a otro

En muchas ocasiones interesa generar un nuevo movimiento partiendo del inmediatamente anterior. Para evitar al usuario el tener que meter todos los valores de movimiento del que se quiere partir, evitando que se produzcan movimientos bruscos y un trabajo repetitivo por parte del usuario.

Para paliar esta necesidad existe este método, que permite sobrescribir un movimiento utilizando como patrón otro. Estos dos movimientos son seleccionables por el usuario, pudiendo elegir el orden en el que se sobrescriben los movimientos. Para optar a esta función basta con enviar el comando `X`. Tras recibir esto, el programa mostrará la misma lista que en la función *Cambiar movimiento*, preguntando al usuario que movimiento desea copiar. Tras recibir el valor correspondiente del vector a cambiar volverá a preguntar al usuario cuál es el movimiento que se quiere sobrescribir. Después de recibir este segundo valor, el robot pedirá una confirmación para ejecutar de forma definitiva este cambio, permitiendo al usuario cancelar esta operación. Si el usuario envía el comando `Y`, se sobrescribirá el movimiento, en el caso contrario de que se envíe cualquier otro valor el programa lo tomará como que se ha cancelado la copia de movimiento. Todo este procedimiento se muestra en la figura 6.21.

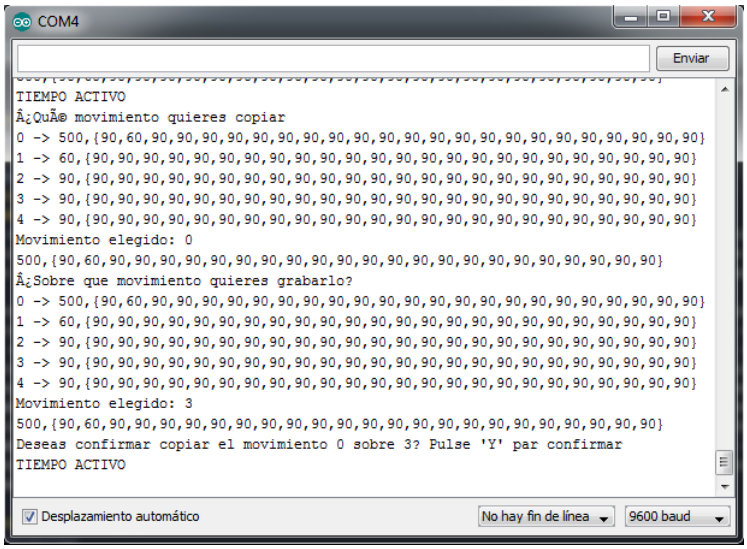


Figura 6.21: Copia de movimientos.

Ejecutar los movimientos

Esta función le da al usuario la opción de probar el movimiento total que ha ido introduciendo hasta ese momento. Esto es de gran utilidad ya que permite al usuario depurar los movimientos y ver el comportamiento real del mismo según los valores que acaba de definir.

Si el robot recibe el comando `E` ejecutará de forma automática toda la matriz de movimientos. El programa irá ejecutando por orden desde el primero hasta el último, basándose el método *move*, cada uno de los movimientos guardados en cada fila de la matriz, respetando la variable del tiempo para cada uno de ellos.

Mostrar todos los movimientos

Esta opción muestra por pantalla todos los movimientos almacenados hasta ese momento de la ejecución del programa, sin necesidad de ejecutarlos. Este método es similar al del programa de *trim*, salvo que en este momento no se muestran los valores de ajuste si no la matriz donde se guardan los movimientos.

Al teclear el comando `S`, aparecerá de forma automática los valores almacenados en la matriz, separando cada movimiento en una línea diferente. A su vez, cada uno de estos elementos está separado entre sí por una coma. Además, salvo el valor del tiempo, el resto de elementos de cada movimiento esta incluido dentro de unas llaves. Este formato es importante por lo que se verá a continuación para poder guardar los datos, y se puede ver en la figura 6.22.

Además la impresión por la terminal de todos los valores, aparece un mensaje de texto que informa al usuario indicándole que copie esa información en un archivo de texto plano para que

luego se pueda procesar para poder usarlo. Como se ha dicho, no se puede guardar el conjunto de movimientos directamente, por lo que este paso es necesario si se quiere poder usar este movimiento una vez que se apague el robot.

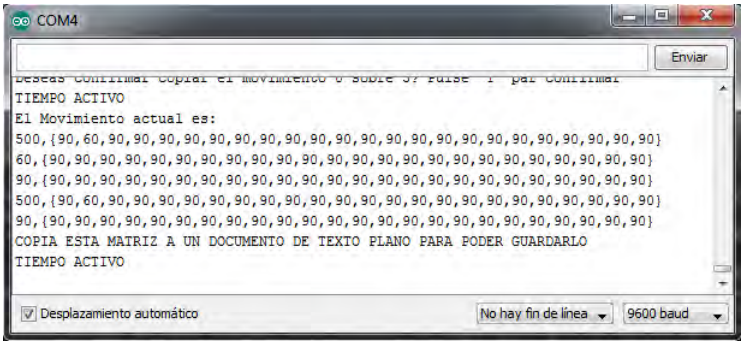


Figura 6.22: El programa muestra todos los movimientos creados hasta ahora.

Ayuda/Mostrar todos comandos

Esta función sirve para recordar al usuario que funciones y que comandos corresponden a cada una de ellas para poder usarlas. Para que aparezca este mensaje es suficiente con enviar el comando `H`, devolviendo el mensaje que se muestra en la figura 6.23.

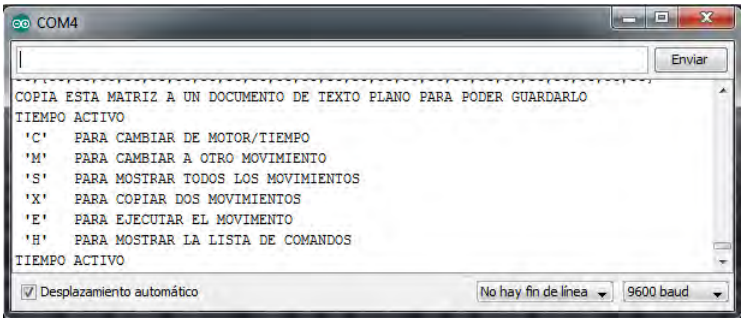


Figura 6.23: Distado de comandos disponibles.

6.6. Conversor de texto a formato de código

En este punto, para poder utilizar los movimiento que se han programado con el *Generador de movimientos por cinemática directa* es necesario transformar su formato en uno similar a que utiliza el *Generador de caminadas*. Este formato, visto anteriormente en 6.4.4, está diseñado para usarlo sobre *Arduino* al incluir lo movimientos en una función de *C++* y con los métodos presentes en la librería *MYOD.h* .

Para esta conversión de los datos guardados en el archivo de texto plano en un fichero con formato se ha creado un programa que automatice esta transformación. Este programa está basado en el lenguaje *C++* y permite leer y escribir ficheros relacionados con el programa visto anteriormente. El funcionamiento de este programa es sencillo, al leer el fichero de texto plano que indique el usuario, almacenar sus datos en variables internas y escribir un nuevo archivo con el nombre de la función que indique de nuevo el usuario. En esta transformación el programa agrupa los datos leídos de tal forma que tengan coherencia para el entorno de *Arduino* para luego poder usarse.

Para usar los datos una vez transformados, basta con volver a seguir los pasos vistos en 6.4.5, al estar guardada la información de una forma idéntica.

El formato de los archivos transformados por este programa y los exportados por el *Generador de caminatas* no son completamente iguales, al existir dos diferencias. La primera de estas diferencias es que no existe la parte relativa a los comentarios que definen la caminata, por motivos obvios al no existir estos parámetros en el *Generador de movimientos*. Y la segunda diferencia es que no aparece una única variable que defina los tiempo. En este caso, cada movimiento tiene un tiempo de ejecución vinculado a él que no tiene que ser el mismo que en otro movimiento. Por este motivo, junto a la declaración de posiciones mediante el vector, aparece una nueva variable que señala el tiempo de ese movimiento.

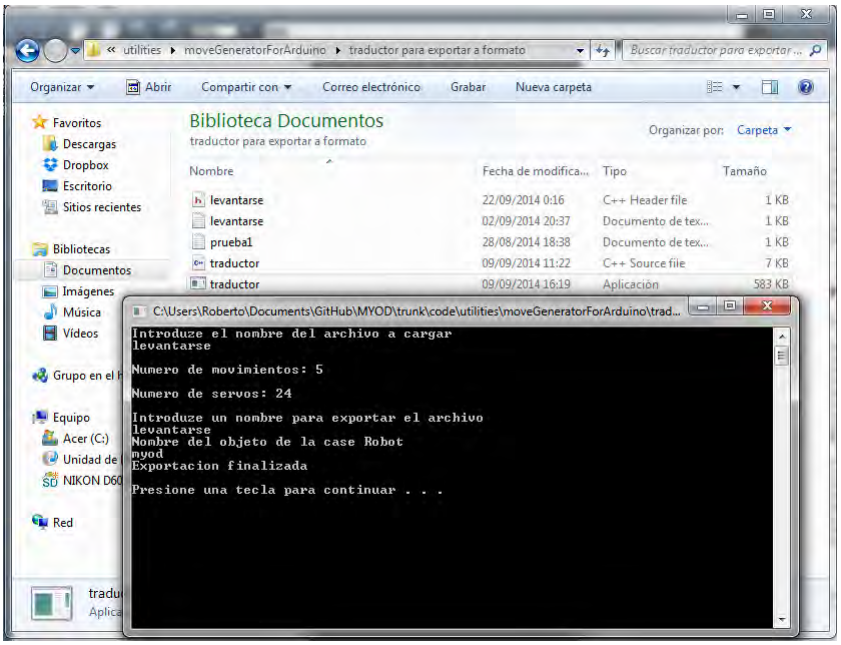


Figura 6.24: Programa Conversor de texto a formato de código.

Tras ubicar el programa en la carpeta donde se encuentre el archivo de texto con la información y ejecutarlo, pedirá al usuario el nombre del archivo a convertir. Si el nombre del archivo no corresponde con un fichero del tipo `.txt`, el programa dará un aviso y procederá a cerrarse automáticamente, como se ve en la figura 6.24. En el caso de que sí exista un archivo con el mismo nombre y extensión, procederá a leerlo, indicando por pantalla el número de movimientos y servos contenidos en ese archivo. Tras esto, volverá a pedir un nombre para exportar el archivo, en esta caso en formato `.h` y el nombre de la clase `Robot` en la que se utilizará el movimiento. Después de introducir estos dos datos, el programa se cerrará automáticamente dejando guardado en la misma carpeta que el fichero exportado.

Capítulo 7

Pruebas Experimentales

Es necesario probar experimentalmente los movimientos, trayectorias y desfases calculado, viendo que combinaciones son posibles y hacen moverse correctamente al robot. En la primera parte de este capítulo se trata de conocer que combinaciones son posibles y cuales desestabilizan el robot. Esto dará una primera idea de que movimientos permiten la estructura y los motores. Tras esto, se intentará optimizar en la segunda parte del capítulo el movimiento de este, ajustando todos los valores presentes en los diferentes movimientos.

7.1. Viabilidad de las trayectorias

Para obtener los valores óptimos que permitan al robot caminar en línea recta se ha realizado un banco de experimentos. En él, se pretenden probar todas las combinaciones posibles usando únicamente funciones sinusoidales y así hallar un rango de valores para cada movimiento. Estos experimentos han de realizarse para cada estudio de balanceo detallado en la sección 4.2 para cada una de las dos versiones.

Estos experimentos están compuestos por otros más simples que hacen a la idea de la viabilidad de los movimientos posibles para este robot. Estos experimentos previos consisten en partir de un único movimiento e ir añadiendo progresivamente el resto de ellos, respetando un orden y los desfases vistos en el capítulo 4.4.

Todos estos movimientos serán creados mediante el conjunto de programas del *generador de movimientos*, visto en el capítulo 6.4. En todos estos experimentos se introducirán dos variables como si fuese constantes. La primera de estas dos constantes es la que define la longitud de movimiento, que será considerada como $166mm$ para todos los casos. Además, la segunda de estas constantes será el número de intervalos en los que se trocea el movimiento, tenido un número de intervalos igual a 40.

En el caso de querer varia la duración total de un movimiento no es necesario generar un nuevo fichero con todos los datos, basta con cambiar el valor de la variable `inc_time` del fichero de salida del *generador de caminatas*. Para calcular el nuevo valor de esta variable basta dividir la duración total del nuevo movimiento entre el número de intervalos, en este caso es un número fijo e igual a 40.

Estos movimientos se utilizarán según lo explicado en 6.4.5, donde la función de caminata generada irá en la parte del programa correspondiente a `loop()`, ya que se quiere que se repita de forma continua e indefinida. Esta función será probada en ausencia de otra, sin incluir ningún tipo de arranque o parada extra en ningún momento del tiempo.

Para las pruebas el robot deberá llevar la batería cargada en la parte que ya se ha indicado y no puede tener ningún cable externo al robot que pueda alterar el equilibrio del mismo. Esto significa que los datos se han de cargar previamente en la controladora con el cable USB y

retirarlo tras ello. Como se ha dicho, la batería ha de cargarse con un elemento ajeno al robot y este ha de llevarla alojada en la parte posterior de la pieza de la cadera.

Como es obvio, el robot deberá estar perfectamente montado y las posiciones enviadas deberán corresponder con las que cada eje alcanza. Para ello, será necesario calibrar el robot usando los valores de ajuste y almacenarlos en la memoria de la controladora. El robot deberá ser totalmente simétrico en la posición inicial de cada eje respecto con su homólogo del lado opuesto.

7.1.1. Pruebas con balanceo

El primer movimiento con el que se va a experimentar es el *movimiento de balanceo*. Este movimiento es el que tiene más estudios que la definen, además de incluir las dos versiones de este primer prototipo. Esta primera tanda de experimentos consistirá en balancear al robot siguiendo una señal sinusoidal con un valor umbral de 0.9 que irá variando la frecuencia de oscilación y la amplitud del movimiento para cada una de las versiones. Es totalmente imprescindible que estos experimentos se realicen sobre una superficie totalmente plana y nivelada, para no obtener datos erróneos.

Para acortar el abanico de posibilidades de estas dos variables, se va a dividir el espacio solución en rangos representativos de los valores cercanos, probando únicamente un valor representativo de ese rango. En primer lugar, para el caso de la frecuencia, se definirán rangos de $200ms$ y para las amplitudes $5mm$. Esto significa que, para simplificar, se elegirán valores múltiplos de los dos números recién mostrados.

Este primer experimento consistirá en balancear el robot, con los brazo estirados y apuntando al suelo, hasta comprobar que todo el peso se reparte en una de las dos piernas. Se considerará como resultado positivo si en una de estas combinaciones el robot separa mínimamente uno de los pies del suelo sin producirse un derribo por el vuelco. Para probar si un pie no soporta nada de peso, se intenta deslizar entre él y la superficie una hoja de papel. En el caso de que esta consiga deslizar en algún momento, se puede considerar que la fuerza de rozamiento es casi nula por no existir peso aplicado en ella y, por lo tanto, que el robot no se apoya sobre ella.

Para homogeneizar estos resultados, es necesario que todos los experimentos se re realicen sobre la misma superficie. Esto se hace para evitar incluir la variable de la adherencia del suelo. En este caso, todos los experimentos se han realizado sobre el suelo del laboratorio.

En los siguientes apartados se muestran en tablas el resultado del experimento para cada movimiento, indicando para que combinación se produce el vuelco. No se va a considerar un rango de valores infinito para experimento ya que, o por capacidad del robot o la de sus motores, es imposible alcanzar esos valores. En general, los motores tiene una limitación de velocidad debida a su inercia y su lazo de control, por lo que no se pueden generar caminatas con un periodo inferior a $600ms$ sin que se produzcan daños en el motor o que estos no alcancen realmente las posiciones indicadas. En el caso de las amplitudes ocurre algo similar, pero el agente limitador en este caso es la estructura. Concretamente las colisiones que se pueden producir entre los elementos l_1 y l_2, por un lado, y l_6 y l_7 por otro. O lo que es lo mismo, que puede colisionar la pierna con la pieza de la ingle y la del pie si además se mueven los ejes que unen estas piezas.

Versión de cadera ancha

Giro de cadera con desplazamiento En este caso, existen algunas combinaciones que permiten que una pierna aguante todo el peso del robot. Como se puede ven en la tabla 7.1, los valores obtenidos se corresponden con un desplazamiento relativamente alto para un robot de estas dimensiones.

Amplitud \ Periodo	600	800	1000	1200	1400	1600	1800	2000	2200	2400
15										
20										
25										
30										
35										
40										
45										
50										
55			•	•						
60			•	•	•	•	•			
65				•	•	•				

Tabla 7.1: Experimento de balanceo con la versión ancha en el movimiento de *Giro con desplazamiento*.

Movimiento horizontal de la cadera Este movimiento también se basa en el acortamiento de una de las piernas para poder mantener la cadera horizontal. Este acortamiento, como se ha visto se consigue rotando q_3, q_4, q_5 y q_6. Por este motivo, existen casos particulares donde se producen colisiones entre elementos al girar los ejes q_3 y q_7 cuando existen desplazamientos altos. Estas colisiones están marcadas con un aspa $\times$ para representar que movimientos no son posibles. Del mismo modo que en el anterior, el rango de valores válidos se encuentra en una región alta de amplitudes del movimientos, como se puede ver en la tabla 7.2

Amplitud \ Periodo	600	800	1000	1200	1400	1600	1800	2000	2200	2400
15										
20										
25										
30										
35										
40										
45			•							
50			•	•						
55			•	•	•					
60	×	×	×	×	×	×	×	×	×	×
65	×	×	×	×	×	×	×	×	×	×

Tabla 7.2: Experimento de balanceo con la versión ancha en el estudio de *movimiento de cadera horizontal*.

Versión de cadera estrecha

Movimiento horizontal de la cadera Como se puede ver en la tabla 7.3, para este experimento los valores aceptables están en una región más extensa, con valores de amplitud entre los $20mm$ y los $45mm$ y periodos comprendidos entre 600 y $2000ms$. Para amplitudes excesivamente altas, el robot es capaz de provocar un vuelco lateral del mismo o una desestabilización notable que lo hace caer en pocos segundos.

Amplitud \ Periodo	600	800	1000	1200	1400	1600	1800	2000	2200	2400
15										
20		•	•	•						
25		•	•	•	•	•	•			
30		•	•	•	•	•	•	•		
35	•	•	•	•	•	•	•	•		
40			•	•	•	•				
45			•							
50										
55										
60										
65										

Tabla 7.3: Experimento de balanceo con la versión estrecha en el estudio de *movimiento horizontal de la cadera*.

Una pierna estirada A diferencia del resto de movimientos, la amplitud en este estudio está definida en grados en lugar de en desplazamiento. Por lo que es necesario cambiar las referencias del experimento. Para este caso, se va a definir una escala para el experimento de 3 grados que se tomará como referencia.

Amplitud \ Periodo	600	800	1000	1200	1400	1600	1800	2000	2200	2400
3										
6										
9										
12										
15	∗	∗	∗	∗	∗	∗	∗	∗	∗	∗
18	∗	∗	∗	∗	∗	∗	∗	∗	∗	∗
21	∗	∗	∗	∗	∗	∗	∗	∗	∗	∗
24	∗	∗	∗	∗	∗	∗	∗	∗	∗	∗
27	×	×	×	×	×	×	×	×	×	×
30	×	×	×	×	×	×	×	×	×	×

Tabla 7.4: Experimento de balanceo con la versión estrecha en el estudio de *movimiento horizontal de la cadera*.

Estos movimientos producen también colisiones entre los mismos elementos que se explica en 7.1.1 y se indican del mismo modo. Además, este movimiento no permite que la totalidad del peso del robot se apoye sobre una pierna en los valores probados. Además, para movimientos con amplitud intermedia, el robot se desestabiliza al no poder levantarse con una sola pierna. Este incidente, que se marca como $*$, se debe a que los motores no son capaces de ejercer tanta fuerza en esta situación, al aumentar el brazo de palanca.

Desgraciadamente, como se puede ver en la tabla 7.4, no existe ningún movimiento válido para este conjunto de valores al usar motores de este tipo.

Ambas piernas encogidas Del mismo modo que para el experimento anterior, este movimiento está definido sobre aperturas y no sobre desplazamientos, por lo que la escala del experimento es igual al anterior.

Amplitud \ Periodo	600	800	1000	1200	1400	1600	1800	2000	2200	2400
3										
6										
9										
12	*	*	*	*	*	*	*	*	*	*
15	*	*	*	*	*	*	*	*	*	*
18	*	*	*	*	*	*	*	*	*	*
21	*	*	*	*	*	*	*	*	*	*
24	×	×	×	×	×	×	×	×	×	×
27	×	×	×	×	×	×	×	×	×	×
30	×	×	×	×	×	×	×	×	×	×

Tabla 7.5: Experimento de balanceo con la versión estrecha en el estudio de *movimiento horizontal de la cadera*.

En este experimento se vuelve a ver que existen amplitudes que provocan la colisión de varios elementos y, en otros casos, la destabilización del robot por la falta de torque que suministran los motores. Al igual que en el caso anterior, no existen combinaciones que permitan un correcto balanceo del robot. Estos resultados se pueden ver en la tabla 7.5.

Análisis de este experimento

Como se ha podido ver en estas pruebas, no están disponibles todas las opciones para balancear cada movimiento y en algunos casos no existe ninguna. Como se acaba de ver, dos de estos cinco experimentos muestran que no se puede balancear el robot siguiendo esos métodos ya que existen problemas notables de diseño y dimensionamiento en la fuerza de los motores. En un experimento adicional se ha podido denotar que el robot no es capaz de levantarse de una flexión de piernas con ángulos mayores a 35º, usando las mismas restricciones que en el estudio *movimiento de rodilla*. En el caso de este experimento adicional, se utilizan ambas piernas para levantar al robot de la postura en flexión. En uno de estos movimientos se encoje una única pierna, pudiendo pensar que al ángulo límite es menor que el del experimento. En el caso del estudio de en el que ambas piernas se encogen, se deben considerar también los esfuerzos dinámicos al variar el peso de una pierna a otra. Además, en este caso las piernas parten de un estado flexionado, por lo que el recorrido hasta el ángulo límite es menor, pudiendo elegir menor variedad de amplitudes.

Esta falta de par no es debida exclusivamente a un mal diseño, si no que también los servomotores no ofrecen tanta potencia como indica el vendedor. En este caso, los motores utilizado para este robot solo son capaces el producir un par aproximado de $7.5 Kg_f \cdot cm$[1].

Por otro lado, si se comparan los otros tres experimentos entre sí, se puede ver que el estudio de basado en la versión de *cadera estrecha* tiene un rango de valores mayores que los otros dos. Esto hace pensar que la longitud de la cadera, a una misma longitud de piernas, varía sensiblemente la estabilidad del robot. Cuanto mayor es dicha longitud, más hay que desplazar el centro de gravedad para repartir el peso. Como es lógico, al existir mayor separación entre apoyos más distancia hay que recorrer desde el punto medio hasta uno de sus extremo.

Por último, siempre el pie que se levanta del suelo con menor amplitud es el mismo, el derecho. Esto denota que el centro de gravedad no está totalmente centrado, si no que está desplazado ligeramente hacia la izquierda del robot. Esto puede ser un problema ya que un pie estará en el aire más tiempo que el otro, generando un movimiento no simétrico haciendo que el

[1]Este dato ha sido obtenido de forma experimental

robot no camine en línea recta. Por el momento no se tratará de resolver el problema, dejándolo para más adelante.

7.1.2. Pruebas de balanceo con movimiento de rodilla

La siguiente tanda de experimentos consiste en ver la estabilidad del robot al estar sustentado únicamente por una de sus piernas mientras que la otra no se encuentra en contacto con el suelo. Para evitar que la pierna no toque el suelo es necesaria encogerla levemente con respecto a su posición natural.

En este experimento se va a intentar usar una señal sinusoidal para encoger la pierna según el *movimiento de rodilla*. De esta señal solo interesará la parte positiva de la misma considerando la parte negativa como nula. Es necesario respetar los desfases calculados en 4.4 para este movimiento con respecto al de balanceo para sincronizar correctamente ambos y producir el efecto deseado.

Para este experimento solo se tomará en consideración los estudios que han superado el experimento anterior, esto quiere decir que se aplicará el *movimiento de rodilla* a los estudios que hayan tenido al menor una combinación viable en el experimento anterior. Por este motivo se descartan los estudios de *una pierna estirada* y *ambas piernas encogidas*.

Además, este estudio se aplicará solo sobre las combinaciones anteriores que han sido exitosas el experimento y las inmediatamente contiguas, restringiendo ampliamente el número de experimentos a realizar.

Para simplificar las variables, la amplitud de este movimiento será de $8mm$, distancia suficiente para ver si existe una separación en entre el pie del robot y el suelo manteniéndose apoyado sobre el otro.

Experimento I

En este experimento se ha visto que en ningún caso se ha conseguido separar los pies del suelo, por comenzar a encoger la pierna mucho antes de que el peso del robot este distribuido sobre la otra. Esto hace que el robot mantenga siempre los pies en contacto con el suelo apoyándose sobre las aristas de este y, en alguno de los casos, llegándose a caer al suelo.

Al no producir ningún caso favorable en ninguno de los tres experimentos, tampoco se muestran las tablas de dichas pruebas ya que no aportarían ningún dato relevante más allá de lo que se explica aquí.

Análisis del experimento I

Se acaba de ver que no se producido ningún resultado favorable en el anterior experimento. Esto puede ser debido a que la trayectoria seleccionada para controlar el *movimiento de rodilla* no es la adecuada. Para este movimiento puede interesar centrar todo el movimiento en los instantes que coincidan con la máxima amplitud del *movimiento de balanceo*, en lugar de hacerlo progresivamente a lo largo del tiempo.

Introducción al experimento II

Visto el nulo éxito que se producido con la señal sinusoidal para el *movimiento de rodilla*, se puede pensar que es necesario cambiar la señal de control. Como se ha dicho, puede interesar seleccionar una señal que concentre todo el movimiento en unos breves instantes del tiempo que coincidan con los valores máximos del *movimiento de balanceo*. Como se ha visto en el capítulo 4.3, existe una señal que se ciña a esto. En este caso, la señal que se va a seleccionar es la señal *trayectoria de pulso*.

Esta trayectoria se define, además de por su frecuencia, por el número que este presente en su exponente. Cuanto mayor sea este número más se concentrará el pulso en torno a su valor máximo. Para conocer el valor aproximado de este exponente se realizará este experimento.

El experimento consistirá, probando para una misma misma amplitud de la rodilla, se intentará buscar que valores son los adecuados para dicho exponente con un valor umbral de 0.9 para la señal de balanceo. En el caso de que este valor umbral disminuya para adecuarse a otro movimiento, el valor del exponente se podrá mantener, al estar definido para unos instantes de tiempo con menor duración que en el caso de que el valor umbral sea menor que 0.9 . No obstante, como se podrá ver más adelante, el valor del exponente puede variar en función del resto de valores umbrales para adecuarse en mayor media a las circunstancias del movimiento. No obstante, este experimento servirá como primera aproximación para conocer unos valores de referencia.

En este experimento se valorará del 0 al 2 la estabilidad presente en el robot. El 0 hace referencia a que el robot es totalmente inestable o que no levanta los pies del suelo. En cambio, el valor 2 hace referencia a la correcta estabilidad y sincronía al levantar de forma continua y periódica ambos pies del suelo. El valor intermedio 1 indica que el robot levanta ambos pies en ocasiones pero se desestabiliza fácilmente, cayéndose en un porcentaje elevado de ocasiones.

Si se analiza la función de *trayectoria de pulso*, se puede ver que esta varía sensiblemente en los primeros valores, realentizándose el cambio según van aumentado. Por esta razón, en la primera década de valores se analizarán los exponentes de dos en dos mientras y a partir de ese valor se irán realizando de década en década hasta un cierto valor. Dicho de otro modo, se analizarán los exponentes 2, 4, 6 ,8 ,10^1, 10^2 y 10^3.

Experimento II

Versión cadera ancha. Giro de cadera con desplazamiento En este apartado sólo se muestran las tablas de los experimentos que incluyen datos en los que existe algún valor distinto a 0. Estas tablas, como las que se ven en 7.6, en las que sí existen datos representativos son las relativas a los exponentes 6, 8 y 10, que se muestran a continuación.

Exponente n=6					
T / Amp	1000	1200	1400	1600	1800
55	0	0	0	0	
60	0	1	1	0	0
65	0	1	0	0	
Exponente n=8					
T / Amp	1000	1200	1400	1600	1800
55	0	0	0	0	
60	1	2	1	1	0
65	0	1	1	0	
Exponente n=10					
T / Amp	1000	1200	1400	1600	1800
55	0	1	0	0	
60	1	2	2	1	0
65	0	1	1	0	

Tabla 7.6: Experimento de balanceo con movimiento de rodilla en la versión ancha en el movimiento de *Giro con desplazamiento*.

Versión cadera ancha. Movimiento de cadera horizontal Del mismo modo que en el estudio anterior, solo se mostrarán las tablas donde aparezcan datos diferentes que cero. En este caso, solo existen dos experimentos, los relativos a los exponentes 8 y 10, con valores representativos, como se ve en la tabla 7.7.

Exponente n=8					
T / Amp	1000	1200	1400	1600	1800
45	0	0			
50	1	1	0	0	
55	0	0	0	0	
Exponente n=10					
T / Amp	1000	1200	1400	1600	1800
45	0	0			
50	1	1	1	0	
55	0	0	0	0	

Tabla 7.7: Experimento de balanceo con movimiento de rodilla en la versión ancha en el *movimiento de horizontal de cadera.*

Versión cadera estrecha. Movimiento horizontal de cadera. Para este caso existen numerosas posibilidades que permiten obtener un el movimiento deseado, con diferentes combinaciones entre sí. En las tablas 7.8 se detallan los resultados de este experimento.

Exponente n=6									
Periodo / Amplitud	600	800	1000	1200	1400	1600	1800	2000	2200
20		0	1	1	1	0			
25		0	1	1	1	0	0	0	
30	0	0	1	1	1	0	0	0	
35	0	1	1	1	1	0	0	0	
40		0	0	1	1	0	0		
45		0	0	0					
Exponente n=8									
Periodo / Amplitud	600	800	1000	1200	1400	1600	1800	2000	2200
20		0	1	1	1	0			
25		0	1	2	1	1	0	0	
30	0	0	1	2	1	1	1	1	
35	0	1	2	2	1	1	1	1	
40		0	0	2	1	1	0		
45		0	0	0					

Exponente n=10									
Periodo / Amplitud	600	800	1000	1200	1400	1600	1800	2000	2200
20		0	1	1	0	0			
25		0	1	2	1	1	0	0	
30	0	1	2	2	1	1	1	1	
35	0	1	2	2	2	1	1	1	
40		0	2	2	1	1	0		
45		0	1	0					
Exponente n=10^2									
Periodo / Amplitud	600	800	1000	1200	1400	1600	1800	2000	2200
20		0	0	1	0	0			
25		0	1	2	1	0	0	0	
30	0	1	1	2	1	1	1	0	
35	0	1	1	2	2	1	1	0	
40		0	1	2	1	1	0		
45		0	1	0					

Tabla 7.8: Experimento de balanceo con movimiento de rodilla en la versión estrecha con el estudio de *movimiento horizontal de la cadera*.

Análisis del experimento II

En estos experimentos se puede ver como la *versión de cadera estrecha* tiene más casos favorables que cualquiera de las otros dos métodos de la otra versión. Para un valor umbral tan limitante como el de este caso, existen pocos casos donde se consiga aunar ambos movimientos. Por este motivo se han ido descartando combinaciones que antes eran válidas para el balanceo.

En el caso de los experimentos para la *versión de cadera ancha*, se ha reducido aún más los casos posibles en los que se consigue combinar estos dos movimientos. De hecho, para el estudio de *movimiento horizontal de la cadera* de esta versión no produce ningún caso favorable. Se ha visto que el robot se destabilizaba en el instante de levantar la pierna, provocando un pequeño vuelco que volvía a apoyar al robot sobre ese mismo pie, sin llegar a alcanzar el punto máximo del *movimiento de rodilla*. Aunque se iniciase el movimiento, este en ningún caso se ha visto que se realizase de forma adecuada el movimiento.

Para el otro estudio de movimiento realizado sobre la misma versión, se han conseguido pruebas en las que sí se consigue combinar estos movimientos sin que se pierda estabilidad.

Si se analizan los datos de la *versión de cadera estrecha* para el *movimiento de cadera horizontal* se pueden ver diferentes combinaciones de los valores hasta ahora vistos. Se puede ver si se comparan las tablas correspondientes con los exponentes 10 y 100, la segunda presenta un menor número de valores positivos, sobre todo en los movimientos más rápidos. Esto se debe a que si se aumenta en exceso el exponente de la función, todos los valores excepto unos pocos presenta valores casi iguales a cero. Los pocos números que no tiene este valor nulo se concentran alrededor del máximo. Si estos números que se reparten a ambos lados del valor máximo son muy pocos, la pendiente generada puede ser muy elevada. Por este motivo y por la inercia de los motores que se utilizan, no se produce un movimiento real de los motores según la trayectoria. Los motores no pueden moverse instantáneamente a la posición que se les indica con esa señal tan breve, únicamente alcanzando un valor intermedio al requerido. Además, estas alteraciones súbitas de posición conllevan una despunte enorme en cuanto al consumo energético del robot. Esto se debe a que los motores demandan una gran corriente

para intentar alcanzar la posición exigida. Esta sobrecorriente, puede dañar los motores, tanto los transistores del *puente H* como el lazo de control del mismo, pudiendo destruirlos o mermar gravemente sus capacidades.

Si se analizan los datos en conjunto, se puede ver que el periodo de caminata que más valores favorables genera es el de 1200 mm para todos los experimentos. Puede dar pistas para un futuro de cuál ha de ser el periodo para la mayoría de caminatas.

Como conclusión de este experimento, se puede ver que los valores del exponente que más compatibilidad producen es encuentran en torno a 8 y 10. Esto significa que se pueden tomar como referencia estos valores para los siguientes experimentos. Concrétamente, se tomará 10 como valor del exponente. Esto no significa que sea el valor definitivo de esta variable, si no que será el valor que se tomará como referencia para los siguientes movimientos.

En este punto se puede descartar el estudio de *movimiento de cadera horizontal* para la *versión de cadera ancha* al no existir ninguna combinación que genere un movimiento acorde con los requisitos. Aunque el otro estudio de la misma versión solo presente unos pocos valores favorables, no se le puede descartar en este momento.

7.1.3. Pruebas de balanceo, movimiento de rodilla y movimiento de avance

En los siguientes experimentos se intentan combinar los tres movimientos fundamentales para el deslazamiento del robot en línea recta. Para reducir el número total de experimentos se partirá de los datos obtenidos en el anterior, donde se ha podido conocer la combinación de valores que produce casos favorables en la oscilación del robot y el exponente que se utilizará para definir la función que controla la trayectoria del *movimiento de rodilla*.

Para denotar de una manera sencilla los resultados del experimento, se indicarán de manera numérica desde el 0 al 3. Se marcará con un 0 los casos en los que el robot caiga al suelo antes de completar dos ciclos de caminata completos, con un 1 si el robot no se cae pero tampoco avanza de manera significativa al estar siempre pivotando sobre sus pies al no conseguir levantarlos del suelo, se marcará con un 2 si el robot consigue desplazarse de la posición inicial pero cae tras un número breve de pasos y con un 3 si el robot se mantiene avanzando de forma continua durante más de 8 pasos sin caerse.

Introducción del experimento I

Para este movimiento solo se tendrán en cuenta los dos estudios que no han sido descartados hasta ahora. En ellos se probarán únicamente trayectorias sobre las combinaciones exitosas. Para la versión de cadera ancha, se probará también en el resto de combinaciones que no han sido completamente favorables pero si han producido un efecto en el movimiento. Esto se hace por no probar una única combinación y asegurarse de que no se pasa por alto ninguna configuración exitosa.

El movimiento de avance se controlará mediante una señal sinusoidal con una valor limitante de 0.9, al igual que el caso anterior referido al *movimiento de balanceo*. Este valor se toma de forma arbitraria con la idea de partir de un valor de referencia. No se considera la opción de utilizar una función sinusoidal pura ya que solo permanece un instante en el punto máximo, siendo difícil la sincronía real con el resto de movimientos.

En este experimento se intentará conocer qué valores para la amplitud del *movimiento de avance* provocan un desplazamiento en el robot. Para no probar todo el abanico de posibilidades presentes, los experimentos se realizarán con incrementos de $5mm$ entre un experimento y el siguiente.

Para denotar de una manera sencilla los resultados del experimento, se indicarán de manera numérica desde el 0 al 3. Se marcará con un 0 los casos en los que el robot caiga al suelo antes de

completar dos ciclos de caminata completos, con un 1 si el robot no se cae pero tampoco avanza de manera significativa al estar siempre pivotando sobre sus pies al no conseguir levantarlos del suelo, se marcará con un 2 si el robot consigue desplazarse de la posición inicial pero cae tras un número breve de pasos y con un 3 si el robot se mantiene avanzando de forma indefinida sin caerse.

Experimento I

Versión de cadera ancha A continuación se pueden ver en la tabla 7.9 los datos conseguidos.

Exponente=10	Balanceo=50				
T / Avance	800	1000	1200	1400	1600
5	0	1	2	1	1
10	0	0	1	1	0
15	0	0	1	1	0
20	0	0	0	0	0
25	0	0	0	0	0
30	0	0	0	0	0

Tabla 7.9: Experimento I con los tres movimientos básicos en la versión ancha en el *movimiento de horizontal de cadera*.

Se puede ver que no existen valores que hagan avanzar de forma correcta al robot, aunque existen combinaciones en los que el robot no llega a caerse avanzando levemente por la fricción al pivotar sobre sí mismo. Aunque se produce un avance neto, no se puede considerar como válido al no estar controlado por los parámetros si no que depende del deslizamiento del robot. Además, se necesitarían numerosos ciclos para que se produzca un avance útil, por lo que no se pueden considerar esos movimientos como validos.

Versión de cadera estrecha En la tabla 7.10 se pueden ver los datos obtenidos para este experimento.

Exponente=10	Balanceo=20						
T / Avance	800	1000	1200	1400	1600	1800	2000
5	0	1	2	2	2	1	1
10	0	0	2	1	1	1	0
15	0	0	1	1	0	0	0
20	0	0	0	0	0	0	0
25	0	0	0	0	0	0	0
30	0	0	0	0	0	0	0
Exponente=10	Balanceo=25						
T / Avance	800	1000	1200	1400	1600	1800	2000
5	0	1	2	2	2	1	1
10	0	0	2	1	1	1	0
15	0	0	1	1	0	0	0
20	0	0	0	0	0	0	0
25	0	0	0	0	0	0	0
30	0	0	0	0	0	0	0

Exponente=10	Balanceo=30						
T / Avance	800	1000	1200	1400	1600	1800	2000
5	0	1	3	3	2	1	0
10	0	0	2	2	1	1	0
15	0	0	2	2	1	0	0
20	0	0	1	1	0	0	0
25	0	0	0	0	0	0	0
30	0	0	0	0	0	0	0
Exponente=10	Balanceo=35						
T / Avance	800	1000	1200	1400	1600	1800	2000
5	0	1	2	2	2	1	0
10	0	0	2	2	1	1	0
15	0	0	1	1	0	0	0
20	0	0	0	0	0	0	0
25	0	0	0	0	0	0	0
30	0	0	0	0	0	0	0
Exponente=10	Balanceo=40						
T / Avance	800	1000	1200	1400	1600	1800	2000
5	0	1	2	2	1	1	0
10	0	0	1	1	1	1	0
15	0	0	1	1	0	0	0
20	0	0	0	0	0	0	0
25	0	0	0	0	0	0	0
30	0	0	0	0	0	0	0

Tabla 7.10: Experimento I con los tres movimientos básicos en la versión estrecha en el *movimiento de horizontal de estrecha*

Se puede ver que existen varios valores que permiten caminar al robot en línea recta. Además, existe numerosas combinaciones que permiten dar algunos pasos al robot sin caerse durante un breve periodo de tiempo.

Análisis del experimento I

En estos experimentos se ha visto que existen combinaciones que hacen andar al robot, por lo menos para una de las dos versiones del robot.

Para los experimentos realizados sobre la versión de *cadera ancha* no se ha conseguido ningún resultado totalmente positivo, aunque existen combinaciones que permiten hacer avanzar al robot de forma estable durante unos pocos instantes. Se podría aprovechar este resultado haciendo caminar al robot durante un breve periodo de tiempo y más tarde pararlo unos instantes para estabilizarlo antes de volver a reanudar la marcha. Desgraciadamente, esta forma de caminar sumada al poco avance que permite el movimiento haría que un breve trayecto durase un tiempo elevadamente alto.

En el caso de la versión de *cadera estrecha*, existen resultados que hacen que el robot avance con fluidez, aunque para un avance muy bajo. Esto permite mover de manera continuada al robot sin tener que pararlo para que se estabilice. El inconveniente vuelve a ser que el avance del robot es muy bajo si se compara con las dimensiones del robot. Esto puede suponer un consumo de tiempo y energía elevados para desplazar el robot una pequeña distancia.

Introducción al experimento II

Se ha visto en los experimentos anteriores existen formas de hacer caminar el robot en línea recta, pero que aparecen en un número muy bajo de combinaciones. En este experimento se intentará aumentar esa número de combinaciones favorables empleando otra trayectoria diferente a la sinusoidal para generar el avance del robot. En particular, se utilizará una onda triangular con un valor umbral de 0.9 .

Al controlar la trayectoria al basarse en esta onda triangular, se pretende generar un movimiento de avance con velocidad lineal constante en el extremo de la pierna, en el caso de que sólo estuviera presente el movimiento de avance.

Se partirá de las premisas de los estudios anteriores para limitar el espacio de experimentos posibles al basar este banco de pruebas en las aproximaciones y resultados que se han obtenido anteriormente.

Experimento II

Versión de cadera ancha En la tabla 7.11 se pueden ver los resultados del experimento II relativos a la *versión de cadera ancha*.

Exponente=10	Balanceo=50				
T / Avance	800	1000	1200	1400	1600
5	0	2	3	3	1
10	0	2	3	2	2
15	0	1	2	1	0
20	0	0	0		0
25	0	0	0	0	0
30	0	0	0	0	0

Tabla 7.11: Experimento II con los tres movimientos básicos en la versión ancha en el *movimiento de horizontal de cadera*.

Al modificar la trayectoria aparecen nuevos valores que permiten mover al robot usando amplitudes mayores, además de nuevos valores que hacen estable la caminata durante breves periodos de tiempo.

Versión de cadera estrecha En la tabla 7.12 se pueden ver los datos experimentales relativos a la *versión de cadera estrecha*.

Exponente=10	Balanceo=20						
T / Avance	800	1000	1200	1400	1600	1800	2000
5	0	2	3	3	2	1	1
10	0	2	3	2	1	1	0
15	0	0	2	1	0	0	0
20	0	0	0	0	0	0	0
25	0	0	0	0	0	0	0
30	0	0	0	0	0	0	0
Exponente=10	Balanceo=25						
T / Avance	800	1000	1200	1400	1600	1800	2000
5	2	3	3	3	2	2	1
10	0	2	3	3	2	1	0
15	0	2	3	2	2	0	0
20	0	0	2	2	0	0	0
25	0	0	0	0	0	0	0
30	0	0	0	0	0	0	0
Exponente=10	Balanceo=30						
T / Avance	800	1000	1200	1400	1600	1800	2000
5	0	2	3	3	3	2	1
10	0	2	3	3	2	1	0
15	0	0	3	2	2	0	0
20	0	0	0	0	0	0	0
25	0	0	0	0	0	0	0
30	0	0	0	0	0	0	0
Exponente=10	Balanceo=35						
T / Avance	800	1000	1200	1400	1600	1800	2000
5	0	2	3	3	3	2	1
10	0	2	3	2	2	1	0
15	0	0	2	2	2	0	0
20	0	0	0	0	0	0	0
25	0	0	0	0	0	0	0
30	0	0	0	0	0	0	0
Exponente=10	Balanceo=40						
T / Avance	800	1000	1200	1400	1600	1800	2000
5	0	2	3	3	3	2	1
10	0	2	2	2	2	1	0
15	0	0	2	2	2	0	0
20	0	0	0	0	0	0	0
25	0	0	0	0	0	0	0
30	0	0	0	0	0	0	0

Tabla 7.12: Experimento II con los tres movimientos básicos en la versión estrecha en el *movimiento de horizontal de estrecha*

Se puede ver que existen un mayor número de posibilidades para hacer avanzar al robot. Además también han aumentado los casos en los que el robot consigue moverse de una forma

estable durante unos breves instantes de tiempo.

Análisis del experimento II

Si se comparan los resultados de los experimentos relativos a las diferentes señales de control para la el *movimiento de avance*, se ve que existen más casos favorables para la señal triangular. En este caso, el número de casos favorables sigue siendo superior para la versión de *cadera estrecha*. Esto puede denotar más estabilidad al tener que balancear el cuerpo en una amplitud menor.

Se sigue viendo que las regiones de tiempo que incluye más casos favorables siguen siendo las relativas a periodos comprendidos entre 1000 y 1400ms, sobre todo para el valor de 1200.

Para amplitudes relativamente grandes, siempre se produce una desestabilización total o parcial del robot, provocando que este caiga casi al instante. No obstante, para las dos versiones se ha producido un aumento en la amplitud máxima que se puede alcanzar sin derribar el robot. Esto puede indicar que se pueden obtener mejores resultados con respecto al la variable *avance*, para ambas versiones.

En contrapartida, ninguno de estos movimientos ha conseguido mantener al robot en pie durante más de quince segundos, siempre provocando una caída hacia atrás del cuerpo. Esta caída es progresiva, empezando a desestabilizarse ligeramente hacia atrás hasta un punto en el que la proyección del centro de gravedad está detrás de la planta del pie que sostiene un peso del robot, generando una aceleración angular que hace caer al robot.

7.1.4. Conclusiones preliminares

Con estos experimentos se ha demostrado que se puede generar un movimiento básico de caminata usando los movimientos de *balanceo, rodilla* y *avance*. Estos experimentos no sirven realmente para cuantificar el valor de las variables que participan en cada ecuación, que se hará en el capítulo siguiente, si no que sirven para demostrar que el robot es capaz de caminar usando la superposición de movimientos y los motores de baja potencia seleccionados por el grupo de trabajo *MYOD*.

También se ha visto que estos motores no son capaces de levantar al robot en un movimiento de *sentadilla* muy pronunciado. Aunque la estructura permita el movimiento en esas regiones, los motores no son capaces de aportar el torque suficiente para levantar al robot. En ángulos superiores a 35 grados aproximadamente, el brazo de palanca que se considera para levantar el robot es mayor al la distancia máxima de palanca que pueden suministrar el robot para el peso del cuerpo y los elementos que se soportan. Esto ya demuestra que los motores están en ocasiones trabajando al límite de su capacidad, pudiendo llegar a dañarse. También pone de manifiesto que estos motores no son los más adecuados para el peso actual del robot, aunque siguen valiendo para poder desplazarle.

Al probar cada experimento, se ha visto que un porcentaje elevado de veces el robot se cae hacia atrás. Esto puede ser debido a la reacción de la inercia ante el avance del robot. Al considerar el robot como un péndulo invertido e intentar avanzar generando el movimiento en la parte inferior de éste, se ve que es relativamente sencillo que el otro extremo pierda en el equilibrio en sentido opuesto por culpa de la reacción. Al no disponer de un inclinómetro o giróscopo, puede ser un gran problema para la estabilidad del mismo, aunque en el siguiente capítulo se explicará como solucionar de una forma sencilla este inconveniente.

En todos los experimentos ha existido un pequeño desplazamiento relativo entre el suelo del laboratorio y los pies del robot, en concreto este desplazamiento se produce justo en los cambios de dirección del *movimiento de balanceo*.

En los últimos movimientos, se ha visto que el robot deriva siempre en el mismo sentido. Esto puede deberse a que el centro de gravedad está desplazado lateralmente, por lo que un pie soporta más peso que el otro. En el caso de que el robot se este moviendo, el centro de gravedad al no estar centrado y se traslada una cantidad igual para ambas direcciones, en una de ellas de el centro de gravedad se alejará más del centro del robot que para la otra. Esta falta equitativa de desplazamiento de gravedad provoca que uno de los pies toque antes el suelo en su recorrido que el otro, haciendo que pivote ligeramente sobre él.

7.2. Optimización de la caminata

En este apartado se realizarán las pruebas para concretar un rango para valores umbrales así como un rango recomendado para cada amplitud de los movimientos. Aunque se ha visto que es viable usar ambas versiones para el movimiento del robot, estos experimentos se centrarán en una única versión. Como se ha visto experimentalmente, la versión *estrecha de cadera* presenta un mayor número de posibilidades, por lo que esta versión será el objeto del experimento. Esto no quiere decir que la otra versión no sea viable, si no que en este proyecto no se pretende analizar en profundidad. Se ha visto que existen combinaciones que permiten mover al robot usando esa versión, pero se presentan un número menor que para la otra en el experimento anterior.

Para el conjunto de movimientos y versión citados se pretende obtener experimentalmente un movimiento totalmente estable y sin desviaciones. Para intentar conseguir la estabilidad se pretende incluir una variable más en el *movimiento de balanceo*. Esta variable servirá como *offset* que permitirá descentrar la trayectoria para compensar la deriva.

En el caso de la estabilidad del movimiento en un número elevado de repeticiones, se intentará desplazar el centro de gravedad hacia adelante. Para ello se jugará con diferentes valores que se añadirán a algún eje de las piernas para inclinar el cuerpo ligeramente hacia adelante. Este desplazamiento del centro de gravedad en el sentido de avance puede resistirse a la inercia producida por dicho movimiento, equilibrando al robot y evitando que se caiga hacia atrás.

Para los siguientes experimentos se utilizarán las mismas consideraciones generales que para los experimentos anteriores.

Por último, cabe destacar que este proceso es iterativo, teniendo que repetir los pasos en el orden que se describen al menos un par de veces. En este caso, solo se realizarán los experimentos una vez, ya que se necesita disponer de tiempo para poder realizar todo el procedimiento correctamente.

7.2.1. Ajustes de los valores limitantes y rango de valores óptimos

En los siguientes experimentos se intentará obtener un rango aproximado de que valores se pueden usar para delimitar cada una de las señales y qué amplitudes dan mejores resultados. Este proceso irá probando diferentes combinaciones de valores para indicar cuales son viables y cuales no. Además, dentro de los caso favorables se indicará qué valores son más adecuados que otros.

En este experimento se fijarán dos de las tres variables que definen el movimiento, variando la otra para recorrer el abanico de posibilidades. Una vez que se haya completado ese abanico, se alterará una de las dos variables fijadas volviendo a repetir el experimento.

Para este experimento, se analizarán los valores para los rangos de 0.5 a 0.9 para los *movimientos de balanceo y avance* y se probarán diferentes valores para la función de pulso encargada del *movimiento de rodilla*.

Se partirá del movimiento que se describe en la tabla 7.13 que se muestra a continuación y se irán probando diferentes combinaciones para ajustar estos valores.

	Amplitud
Balanceo	25
Rodilla	10
Avance	10
Versión	Cad. Estrecha
Movimiento	Cad. Horizontal

Tabla 7.13: Configuración experimento de inclinación.

Tras obtener el valor de estos parámetros, se procederá a obtener qué amplitudes dan mejor resultado. Este procedimiento puede ser iterativo, repitiéndose tantas veces como se desee hasta la convergencia de los valores. En este caso, solo se realizará una vez, ya se tarda un tiempo considerable en repetir cada tanda de experimentos.

Experimento para los valores umbral

La primera variable que se fijará será la rodilla. Este parámetro tomará valores 8, 10 y 12, generando tablas diferentes. Dentro de cada tabla, se probarán valores separados entre sí 0.1, para los umbrales de los otros dos movimientos.

Es difícil clasificar la calidad de las diferentes combinaciones y casi todas ellas provocan movimientos estables. No obstante, existen movimientos que son más estables que otros. Esto se puede observar al que son más fluidos y que apenas presentan movimientos bruscos o generan impulsos no deseados a lo largo de la pierna. Por lo tanto, estos experimentos se pueden diferenciar entre sí clasificándolos desde el 0 al 3, siendo 0 una combinación que provoca una caída y 3 un movimiento totalmente fluido.

Exponente=8					
Avance \ Balceo	0.5	0.6	0.7	0.8	0.9
0.5	2	2	2	2	1
0.6	2	2	2	1	1
0.7	2	3	2	1	1
0.8	2	2	1	1	0
0.9	2	2	1	1	0
Exponente=10					
Avance \ Balceo	0.5	0.6	0.7	0.8	0.9
0.5	2	2	2	2	2
0.6	2	2	2	2	2
0.7	3	3	3	2	1
0.8	3	3	2	1	1
0.9	2	2	1	1	1

Exponente=12					
Balceo / Avance	0.5	0.6	0.7	0.8	0.9
0.5	2	2	2	2	1
0.6	2	2	2	1	1
0.7	2	3	2	1	1
0.8	2	2	2	1	1
0.9	2	2	1	1	1

Tabla 7.14: Experimentos para los parámetros umbrales.

Se puede ver que que la tabla 7.14 que más datos favorables presenta es la relativa al exponente con valore igual a 10. Existen gran número de combinaciones que mueven al robot, pero solo las indicadas con el valor 3 parecen que provoca menores variaciones de velocidad en los límites. Interesa que los esfuerzos sean lo menor posible para evitar que los motores tenga que ofrecer más potencia para resistirse a ellos.

Dentro de los valores que parecen mejor, los mejores parecen ser los que presentan valores más altos. Esto se debe a que el cambio de trayectoria que se producen en el punto de corte es menor. Si se sigue este idea, es fácil razonar que es mejor elegir una pareja de valores que limiten las señales en puntos lo más altos posibles, ya que se sigue produciendo un movimiento fluido con menores variaciones en la trayectoria.

Por lo tanto, es lógico elegir la combinación de valores que mayor tenga los umbrales. Si volvemos a los datos de este experimento (tabla 7.14), se puede ver que la combinación que produce un movimiento fluido con los valores más alto se corresponde con los datos que se muestran a continuación en la tabla 7.15.

Exponente	10
Lim. Balanceo	0.6
Lim. Avance	0.8

Tabla 7.15: Valores umbral con mejor resultado.

Experimento para las amplitudes de los movimientos

Una vez que se han obtenido los valores umbral que definen cada señal, es necesario volver a recalcular las amplitudes de esas trayectorias para intentar mejorar el movimiento del cual se parte.

En este caso se intentará probar diferentes amplitudes, basándose en los valores umbral recién calculados, para conseguir un movimiento de caminata estable y con una velocidad de avance moderada. Para ello, se han de probar diferentes valores de forma experimental para ver cuales parecen más adecuados.

Los experimentos consistirán en ir probando diferentes combinaciones de amplitudes y tiempos, marcando el resultado de dentro de una escala numérica. Se numerará desde el 0 al 3 el resultado de la estabilidad del robot durante se ejecuta el movimiento. Es difícil en este punto discriminar que combinaciones son mejores que otras, ya que los movimientos están en en un grado de depuración cada vez mayor.

Se probarán diferentes combinaciones basadas en múltiplos de 3, ya que no se desean probar todas las combinaciones ya que si no la fase de experimentación se alargaría innecesariamente en el tiempo. Se probará también diferentes tiempos para saber que velocidad de movimiento es la más adecuada, pero en este caso el intervalo entre una y otra prueba es de 200 milisegundos.

En concreto, en la siguiente tabla se muestran los valores de los experimentos relativos a un tiempo de movimiento de 1200 ms ya que es el intervalo que mejores resultados ha dado, aunque se ha probado con valores de 800, 1000, 1200, 1400 y 1600 milisegundos, aunque no se mostrarán en detalle. Tras el movimiento de 1200, cabe destacar la siguiente mejor tabla de tiempos corresponde a 1000 ms.

Rodilla=9	Tiempo= 1200ms				
Avance \ Balceo	18	21	24	27	30
9	1	2	3	3	2
12	1	2	2	2	2
15	1	1	2	2	2
18	0	1	1	1	1

Rodilla=12	Tiempo= 1200ms				
Avance \ Balceo	18	21	24	27	30
9	1	2	3	3	2
12	1	2	2	3	2
15	1	1	2	2	2
18	0	1	1	1	1

Rodilla=15	Tiempo= 1200ms				
Avance \ Balceo	18	21	24	27	30
9	1	2	2	2	2
12	1	2	2	2	2
15	1	1	2	2	2
18	0	1	1	1	1

Tabla 7.16: Experimentos para las amplitudes de los movimientos.

Como se puede ver en los datos de este experimento reflejados en la tabla 7.16, existen movimientos que parecen mejor en comparación a otros. No obstante, como se ha dicho, todos los movimientos son válidos, pero en este punto se ha de elegir el que parezca mejor.

Para elegir el mejor movimiento, es mejor definir previamente los criterios que se van a seguir. Ante todo, interesa la estabilidad, que se ha definido de forma numérica con la escala que se ha usado para este experimento. En segundo lugar, interesa que el robot avance la mayor distancia posible en en menor tiempo posible. En la tabla 7.16, está definida para un movimiento de 1200 milisegundos, por lo que se ha de seleccionar el valor mayor de amplitud. Y por último el consumo energético, concrétamente el instantáneo. Cuanto mayor sea la velocidad de las articulaciones, mayor será el consumo según la fórmula $P = \omega \cdot \vec{M}$. Al aumentar la velocidad, también lo hacen los efectos dinámicos por la inercia, por lo tanto, los dos términos que influyen en la potencia aumenta, aumentando el consumo energético instantáneo del robot. Por esta razón, además de producir movimientos algo más estables, los movimientos lentos consumen menos potencia y por eso solo se muestran los resultados del movimiento relativo a 1200 ms en vez de hacerlo también para 1000 ms, ya que es un movimiento más rápido y de mayor consumo.

Se puede ver, siguiendo este criterio, que el movimiento que mayor amplitud presenta con un movimiento estable es el caso que se muestra en la tabla 7.17.

Movimiento	Amplitud
Balanceo	27
Rodilla	12
Avance	12

Tabla 7.17: Resultado final del experimento.

7.2.2. Ajuste de valores de cintura y brazos

Tras definir y optimizar el grueso del movimiento, es el momento de definir otras características secundarias, como el movimiento de brazos y el de cadera.

Estos movimiento realmente no influyen excesivamente en el movimiento, salvo que se elijan amplitudes excesivamente altas que desestabilicen el robot. En este caso, no se intentará optimizar la caminata optimizando estos movimientos, simplemente se darán unos valores aproximados para que emulen en mayor medida el movimiento humano, balanceando los brazos y moviendo levemente la cintura.

No se impondrán valores limitantes para este movimiento, queriendo que sean lo más fluidos posibles ya que, exceptuando los desfases, no necesitan estar totalmente definidos y sincronizados con otros, siendo casi independientes. Cuanto más fluidos sena estos movimientos, menos esfuerzos dinámicos puntuales tendrá que resistir el robot.

En este caso, de forma experimental y sin ningún criterio estricto, se ha decido que las dos posibles amplitudes que no alteran mucho la estabilidad del robot son los que se muestran a continuación (tabla 7.18).

Movimiento	Amplitud
Brazos	12^o
Cintura	5^o

Tabla 7.18: Valores para los movimientos de brazos y cintura.

7.2.3. Ajustes de inclinación

Tras ajustar los valores limitantes para cada trayectoria y el rango de valores óptimos, es siguiente paso es hacer que el robot ande de manera estable durante más ciclos de caminata.

Se ha visto que el robot tiene a caerse de espaldas una vez que lleva ejecutando el movimiento de caminata un número medio de ciclos. Si se desplaza el cuerpo hacia adelante, el centro de gravedad lo hará con él, pudiendo disminuir el par desestabilizante. Si el cuerpo del robot se inclina demasiado, se puede provocar el efecto contrario, haciendo caer el robot hacia adelante.

Para inclinar el cuerpo hacia adelante se pueden girar los ejes q_3, q_4, q_5 y q_6 de cada pierna una cierta amplitud en el sentido correcto. En este caso, aunque los otros dos ejes de la pierna lo permitan, únicamente se va a analizar los giros en q_3 y q_7, que corresponden a las articulaciones del tobillo y la ingle de cada pierna.

En este apartado se intentará obtener unos valores para estos cuatro ejes, dos en cada pierna que permitan una mayor estabilidad del robot sin tener que recurrir a un lazo de control ni sensores de inclinación.

Todos los ángulos que se definan se consideran desde la vertical del robot, que debe coincidir con la posición inicial de cada motor si se ha ajustado correctamente. No se detallará el sentido exacto del giro de cada servo, analizándose las dos piernas en conjunto.

Estos giros se irán modificando en el *trim* del robot, sin alterar de momento las trayectorias del *generados de caminatas*. Para realizar cada experimento, se ha de ajustar de nuevo el *trim* con el programa homónimo diseñado para este proyecto y visto en 6.2.

Rango máximo de valores

El primer experimento consiste en probar combinaciones de giro para las articulaciones de los tobillos e ingles para un movimiento predeterminado. En este caso, se utilizará uno de

los movimiento que mejor resultado han dado incluyendo sus mismos valores umbral y sus amplitudes características.

	Amplitud	Umbral
Balanceo	27	0.6
Rodilla	12	10
Avance	12	0.8
Brazos	12	×
Cintura	5	×
Versión	Cad. Estrecha	
Movimiento	Cad. Horizontal	

Tabla 7.19: Configuración experimento de inclinación.

En primer lugar, es necesario indicar como se van a variar cada valor. Para ello, se harán dos barridos diferentes, uno para la inclinación de los tobillos y otra para las ingles. Si se supone que el centro de gravedad está en alguna región cercana a la cadera, se puede ver que la distancia a ese punto medida desde los tobillos es mucho mayor que si se midiera desde las ingles. Esto significa que una variación en los tobillos influye en mayor mediada en el desplazamiento del centro de gravedad que si se provocase en la parte superior de la pierna. Por ello, el primer barrido se hará sobre los tobillos y dentro de este se realizará un segundo barrido moviendo las ingles. En cualquier caso dentro del barrido de ingles se aumentará el valor del ángulo, reiniciándose al iniciar un nuevo barrido. Esto además provocará el incremento del ángulo que han de girarse los tobillos.

Experimentalmente se ha visto que un giro aproximado de $3grados$ en la ingle provocan un efecto similar que si se girase uno en la ingle. Esto define el número máximo en el barrido de la ingle.

Tras realizar un primer experimento se ha visto que un valor aproximado de 4 grados en el tobillo provoca que el robot se caiga hacia adelante al ejecutar el movimiento de caminata, por lo que en los siguiente análisis no se sobrepasará ese valor.

Datos Experimentales

Tras los experimentos, se ha visto que los valores óptimos se han hallado para distintas combinaciones, y todos se muestran a continuación en la tabla 7.20.

Tobillo	Ingle
2	2
2	3
2	4
3	0
3	1

Tabla 7.20: Valores favorables del experimento de inclinación.

Cabe destacar que los valores medios son más estables que los extremos. Los valores extremos hacen caer al robot en determinados casos en los que el firme presenta alguna irregularidad, como una junta entre baldosas. Se recomienda usar los valores intermedios, en concreto [2,3], ya que es el que mejores resultados ha obtenido en la estabilidad del robot al pasar sin destabilizarse sobre pequeñas irregularidades.

Es importante destacar que estos valores dependen exclusivamente del movimiento del robot y los elementos que porte, ya que esto segundo desplazará el centro de gravedad del robot. Estos experimentos solo indican los pasos a seguir para obtener los valores para cada caso y no se han de tomar como datos que siempre se han de considerarse como válidos. Además, aunque se mantenga la configuración de elementos presentes en el robot que definen el centro de gravedad, los valores pueden cambiar además por el movimiento de caminata que se use, siendo el parámetro más influyen la amplitud del *movimiento de avance*.

7.2.4. Ajustes del centro de gravedad

Tras obtener un movimiento de caminata estable, es necesario ajustar el centro de gravedad lateralmente para conseguir una caminata lo más recta posible. En raras ocasiones el centro de gravedad del robot coincide con el plano central de robot, que define la simetría del mismo. Elementos como la electrónica, los conectores o la batería pueden desplazar el centro de gravedad lateralmente. Esto puede ocasionar problemas en cuanto al direccionamiento del robot. Como se ha explicado anteriormente en 7.1.4, un desplazamiento lateral del centro de gravedad provoca que los tiempos en los que los pies permanecen apoyados no sean iguales entre sí. Esto hace que durante un breve periodo de tiempo el robot pivote sobre un pie cambiando la orientación de todo el cuerpo. Para evitar esto se debe considerar el centro de gravedad en la señal que controle el *movimiento de balanceo*, desplazando su ecuación del origen. Para lograr este desplazamiento basta con sumar un valor constante a dicha función.

En esta fase de los experimentos se intentará buscar un valor aproximado para contraponerse a la desviación provocada por el centro de gravedad.

Para poder realizar estos experimentos es necesario hacer modificaciones sobre el programa *generador de caminatas*, concrétamente sobre las partes relativas a la generación de la trayectoria de balanceo. La modificación que hay que hacer es la siguiente.

$$desp = desp_{max} \cdot f(t) + cdg$$

En esta ecuación de ha incluido un nuevo término que antes no aparecía, haciendo referencia al desplazamiento del centro de gravedad cdg y sus unidades son mm. Este término suma un valor constante al desplazamiento para ese instante de tiempo, que luego servirá para calcular mediante las ecuaciones correspondientes al balanceo el giro de cada eje. El inconveniente de usar este método reside en que para un valor $f(t) = 0$ el robot no estará en una posición centrada, aunque el centro de gravedad sí lo esté.

De forma iterativa, se pretende obtener un valor aproximado que haga que el robot avance lo más recto posible. Dependiendo para que lado se desvíe se irá aumentando o disminuyendo esta variable, pudiendo llegar a obtener valores negativos.

Resultado del experimento

El valor obtenido tras diferentes pruebas ha sido que el valor del centro de gravedad es de -2.5mm. Con este valor, y para esta configuración del robot, se consigue que el robot camine totalmente recto. Se considera así ya que en una distancia de 80 cm el robot no se ha desviado más de 5 cm.

7.3. Otros movimientos

En este apartado se pretende, no tanto como dar valores exactos, probar los movimientos de los que se han hablado en la sección 4.5. Se pretende ver la viabilidad de los movimientos y

no sólo indicarlos de forma teórica, ya que este proyecto está orientado a un movimiento real y práctico del robot analizado.

7.3.1. Movimiento de arranque y parada

En este apartado se pretende conocer qué movimientos son adecuados para mover el robot en los instantes que se inicia o se finaliza el movimiento de caminata.

Basándose en los datos obtenidos en la *optimización de la caminata*, que servirán como referencia para estos dos movimientos, se pretende hacer un breve análisis de la utilidad de estos movimientos para el robot y su estabilidad. Para ello, se utilizarán los valores que definen las trayectorias que se muestran en la tabla 7.19, así como el valor del desplazamiento del centro de gravedad, visto en 7.2.4, y se han de usar los mismos valores de *trim* que permiten inclinar levemente al robot.

Movimiento de arranque

En primer lugar, se analizará este movimiento inicial. En este experimento se intentará ver el sentido práctico de lo visto en los cálculos teóricos, decidiendo que movimientos son más adecuados para el propósito de la estabilidad.

Se ha visto, que al aplicar valores limitantes muy restrictivos al *movimiento de balanceo*, por debajo de 0,7, se produce un tiempo demasiado largo en la que los extremos de este movimiento. Al detenerse en estos puntos, es casi innecesario el balanceo previo, ya que no se produce el efecto que se busca, al querer un aumento progresivo en la inercia. Al existir estas paradas duraderas en los extremos, la fluidez se pierde, siendo muy poco práctico este movimiento.

En el mejor de los casos, puede interesar balancear solo durante medio periodo, iniciando el balanceo en el sentido opuesto al que realizará el primer paso. Para generar este primer balanceo, se puede usar el *Generador de caminatas* y modificar el archivo final para ceñirlo al problema. Para ello, basta con introducir el valor del parámetro de balanceo, longitud de la pierna y los otros parámetros de duración de movimiento y número de intervalor, introduciendo en el resto de valores 0. Al definir el *movimiento de rodilla y de avance* como nulos, el programa no incluirá en el movimiento final a estos dos estudios. Tras guardar el archivo, al querer solo medio periodo, es necesario borrar los primeros valores hasta producir el movimiento deseado. El valor que hay que borrar es hasta el paso por cero. Esto quiere decir que se han de borrar la primera mitad de los movimientos.

Por otro lado, se ha visto que un primer paso que se da es el que más desestabiliza al robot. En este caso, si que es necesario, para evitar esto, dar un primer paso con menor valor de avance. Eligiendo un valor que sea la mitad del movimiento de avance que se use en la caminata, se consigue un inicio estable.

Se ha visto, que si no se utilizan estos dos métodos, el robot no llega a caerse, pero varía levemente la orientación que tiene, cambiando la dirección en la que se va a desplazar el robot. Por lo que se aconseja que se utilice dos inicios secuenciales: un *medio balanceo* y un primer ciclo con un avance la mitad del que se utilice en el movimiento de caminata.

Movimiento de parada

Se ha visto experimentalmente que no es necesario un movimiento adicional para parar el robot. La postura base del robot es muy estable, si en ella se da el tiempo suficiente para que absorba todos los esfuerzos dinámicos hasta detener por completo al robot. Esto quiere decir que, si no se ejecuta otro movimiento casi instantáneamente tras detener el robot, éste será capaz de no caerse.

No obstante, se puede utilizar un método similar al anterior para que el robot no pierda la orientación al detenerse, incluyendo dos movimientos que le detenga de una forma más estable. Para ello, se puede utilizar el mismo movimiento de *paso corto* y después un medio balanceo que detenga el avance en dos etapas. A diferencia que el movimiento de arranque, la parte de código que hay que borrar es la segunda mitad de fichero de movimiento de balanceo, ya que interesa que no se produzca salto o movimiento rápido de los motores.

7.3.2. Movimiento de desplazamiento Lateral y Giro

Estos movimientos que definen la totalidad del movimiento del robot no serán creados con el *Generados de caminatas*, si no que se harán con el programa *Generados de movimientos por cinemática directa*. En este programa es necesario definir todos valores de las articulaciones en cada intervalo para provocar el movimiento final. Como se ha visto, se pueden utilizar alguno de los cálculos hechos hasta ahora para orientar los movimientos de una forma más practica. Es el caso de las restricciones y análisis que se han impuesto en el *movimiento de rodilla*, donde todas las articulaciones que aparecen en ese movimiento giran un mismo ángulo. Por ello, al simplificar el movimiento, se utilizará esta consideración al levantar el pie del suelo.

Giro

En primer lugar, este movimiento se ha desarrollado siguiendo las pautas que se dan en el apartado 4.5.2, creando unas trayectorias aproximadas que se asimilen en el caso teórico. Para saber el ángulo girado en las articulaciones para el desplazamiento de la cadera, se puede recurrir a un movimiento creado con el *generado de caminatas*, pudiendo ver los valores exactos para un desplazamiento que bascule el peso del robot.

Se ha visto que este movimiento logra el objetivo deseado, al hacer rotar el robot sobre sí mismo sin que se produzca un desplazamiento significante. Además, el robot pivota levemente sobre el pie favoreciendo el giro.

Por otro lado, también se ha probado el giro apoyándose en la inercia que se genera al mover la cintura rápidamente. Se ha podido comprobar que ese giro es estable, aunque en un pequeño porcentaje de ocasiones el robot se cae al suelo. Por lo tanto, por su gran simplicidad y rapidez, es más aconsejable utilizar esta forma de giro.

Desplazamiento lateral

Por último, este movimiento se puede obtener haciendo unas modificaciones al movimiento de balanceo del movimiento de caminata, si se hace nulo el valor de avance. En el momento que el robot levante el pie, se puede girar los ejes de la cadera para separar las piernas. Además, para mantener el pie con la misma orientación, quedando siempre paralelo al suelo, es necesario girar levemente el pie el mismo ángulo girado en la cadera, para que cuando aterrice en el suelo es apoyo sea firme. Una vez que se llega a este punto, se puede devolver al robot a su posición inicial.

Se ha visto que el robot desliza levemente sobre el suelo, provocando un desplazamiento menor al deseado. El desplazamiento lateral es muy pequeño, teniendo que repetir el proceso varias veces para producir un movimiento práctico.

Capítulo 8

Conclusiones

Mediante los cálculos previos y los experimentos se pueden extraer numerosas conclusiones sobre el robot, tanto de la estructura, electrónica, programación y actuadores. Se ha visto a lo largo del proyecto las ventajas e inconvenientes que presentan cada elemento que participa en el robot, por lo que es necesario dejarlas anotadas para poder partir de un estado más avanzado para la siguiente versión a desarrollar en el futuro.

En los siguientes apartados se mostrarán qué conclusiones se puede sacar después de un uso práctico y real del robot, así como de todas las incidencias que han aparecido a lo largo del proyecto que no se han detallado hasta ahora. Por otro lado, se pretende dejar por escrito una lista de desarrollos futuros y mejoras que permitan hacer avanzar a este proyecto.

8.1. Desglose de conclusiones

En líneas generales, se puede decir que este proyecto presenta resultados favorables, al haber conseguido desplazar al robot por el plano de las formas deseadas al inicio del éste. A lo largo del proyecto se han presentado nuevas complicaciones no previstas al inicio del mismo que se han intentado solucionar de la forma más sencilla y eficiente posible, adaptándose a los recursos disponibles en ese momento.

A continuación se va a detallar por apartados las diferentes conclusiones que se han realizado dependiendo del área al que pertenecen.

8.1.1. Actuadores

En primer lugar, los servomotores seleccionados para la movilidad del prototipo trabajan casi al máximo de sus capacidades en todo momento. Esto no los descarta directamente como posible solución al problema de la movilidad del prototipo, pero tampoco son los más adecuados para mover de forma correcta al robot.

En primer lugar los motores *TowerPro MG996r* no dan la potencia suficiente para mover de una forma totalmente controlada al robot. Si se descompone la ecuación de la potencia en otros dos términos $P = \omega \cdot \vec{M}$, se puede ver que está se formada por velocidad angular y el par que se ejerce en ese momento. En muchas ocasiones, los servomotores presentan un estado transitorio al iniciar un movimiento que los aleja durante unos instantes del trayectoria real demandada. Esto se debe a la inercia que han de vencer al iniciar el movimiento. Cuantos más elementos tengan que mover, más inercia tendrán que vencer para iniciar el movimiento. Esto ocasiona que al arrancar uno de estos motores, se produzcan pequeñas desetabilizaciones al no ceñirse a la trayectoria indicada. Esto es debido a la falta de par instantáneo que el motor puede producir para salir de este estado transitorio.

Por otro lado, estos motores presentan ciertas holguras que no permiten alcanzar de forma exacta la posición. Aunque tiene engranajes metálicos, los brazos que transmiten el movimiento

son de plástico, siendo uno de los responsables de esta holgura. Además, la holgura se produce por la falta de precisión del potenciómetro que se utiliza como encoder analógico. Por último, el mayor causante de esta holgura se produce por el montaje mecánico. Si el tornillo que aprieta el brazo transmisor contra el piñón del servo no ejerce la fuerza suficiente, provoca una holgura mayor que la provocada por los dos efectos anteriores. Para reducir esta última holgura basta con apretar los tornillos de los piñones de cada servo. Esto es tiene una gran importancia por lo que se recomienda revisar estos elementos antes de arrancar el robot para poder conseguir unos movimientos exactos.

Otro problema que aparece es el del consumo energético de los motores, tanto en conjunto como por separado. Si se vuelve a analizara la ecuación $W = \omega \cdot \vec{M}$ se puede ver que la potencia requerido por el motor aumenta según lo haga la velocidad de giro o el par que ha de ejercer. Suponiendo una velocidad uniforme, en los estados transitorios el motor ha de suministrar un torque elevado para poder vencer a la inercia lo más pronto posible. Este aumento puntual de torque hace que la demanda energética del motor se dispare unos instantes. La potencia que se demanda ha de salir de la fuente de alimentación, en este caso de la batería y el regulador de tensión. Si la iguala la potencia eléctrica a la potencia de demanda el motor se obtiene $V \cdot I = \omega \cdot \vec{M}$. Si se considera que el lazo de control del *UBEC* es lo suficientemente rápido y capaz de estabilizar la señal casi instantáneamente, se puede considerar el voltaje como constante. Esto quiere decir, que ante un estado transitorio, el torque del motor aumenta, haciendo que la intensidad lo haga con él. Esta sobre intensidad instantánea puede dañar los elementos que controlan al motor de continua, como los transistores del *puente H*. Estos elementos son los más vulnerables ante una sobre corriente, pudiendo llegar a quemarse en uno de estos picos o en una sucesión repetida de ellos.

En muchos de los movimientos se producen cambios de velocidad casi de forma constante, lo que produce estados transitorios de todo momento. Esto provoca que en casi todo momento los motores consuman más corriente, llegando a consumos cercanos al medido para el estado transitorio. Al tener que mover una carga, la potencia necesaria es mayor. Si se unen estos dos fenómenos, el consumo real del motor es mucho mayor al calculado nominalmente. Se se tiene en cuenta que se están moviendo en torno a más de una quincena de motores en el movimiento de caminata, la intensidad total demanda es elevada. Esto significa que el regulador de tensión ha de hacer frente a corrientes mayores a las calculadas, pudiendo operar casi en la totalidad del tiempo en una corriente superior a su calor nominal. De este problema se darán más detalles más adelante.

Otro problema que conlleva el uso de estos motores es que no se puede saber la posición real en la que se encuentra el motor. El mayor problema de esto se encuentra al encender el robot. En el primer instante de tiempo tras arrancar la placa no se conoce la posición en la que se encuentran los motores, no obstante, se envía la primera orden de movimiento para llevarlo a la posición inicial. Esto conlleva un movimiento en las que se fija las posiciones finales sin conocer las posiciones iniciales. Esto provoca que los motores se muevan rápidamente a esta posición haciéndolo a máxima velocidad y sin ningún tipo de control. Esto puede ocasionar que diferentes piezas colisionen a alta velocidad, pudiendo provocar daños graves a la estructura o a los motores. Además este arranque brusco aumenta repentinamente la corriente demandada.

La falta de realimentación que ofrecen estos motores origina más problemas. En primer lugar, no se puede conocer la posición real que alcanzan los motores, teniendo que hacer aproximaciones y suposiciones para realizar los diferentes cálculos que se llevan a cabo.

Una complicación añadida de estos motores, si se comparan con cualquier motor utilizado en los robots comerciales que se han visto en 3.2, es que no presentan ningún tipo de controladora local dentro de cada motor. Esto obliga tener que procesar toda la información dentro de la controladora, sin poder liberar parte de este trabajo al repartirlo entre los motores. Por ejemplo, si se usase los métodos relativos a la matriz jacobiana inversa para mover el robot, se podría dar

la orden directa de la velocidad de giro de ese motor durante ese intervalo de tiempo ejecutando la orden el motor y haciendo sus propios cálculos de posición para cada instante del intervalo. Al no disponer de esto, es necesario procesar la trayectoria previamente, pudiendo indicar únicamente al motor la posición instantánea. Por otro lado, este motores tampoco disponen de registros ni sensores de los cuales poder leer el consumo, el par o la posición instantánea de cada motor. Esta información extra podría haber decantado al proyecto a unas líneas de desarrollo diferentes, al poder usar métodos de control a tiempo real.

Para finalizar con los actuadores es necesario decir que se ha tenido que reparar en numerosas ocasiones el robot al romperse algún motor. Esto esta provocado en la mayor parte de los casos por pedir al motor pares que no puede suministrar, rompiendo la electrónica que le controla. Si se bloquea el motor por culpa de que este no puede vencer a las fuerzas que intenta vencer, parecen intensidades que el motor no puede soportar. Un motor de continua está compuesto por diferentes devanados, que al fin y al cabo son bobinas enrolladas. Sin entrar en características concretas, el *rotor*, la parte móvil del motor solidaria al eje de giro, se mueve ya que se se activan diferentes devanados que crean un campo magnético que se opone al generado por el *estator*, provocando el movimiento. Para generar estos campos, es necesario hacer pasar una corriente por estos devanados. En el caso de que se produzca el bloqueo, y volviendo a la imagen 2.12, el devanado activo será siempre el mismo. Las bobinas, en el caso de estar alimentadas a una tensión constante y haber superado el estado transitorio, se pueden considerar como un cortocircuito. Esto hace, al existir una baja impedancia, que existe una corriente excesivamente alta.

Este cortocircuito provoca casi instantáneamente la destrucción del *puente H* y del sistema de control. Además, se ha visto experimentalmente que cuando esto ocurre se deterioran varios motores a la vez, al cortocircuitar el voltaje nominal del robot con la masa del circuito, también pudiendo dañar el regulador, la batería o la controladora. Esto provoca un efecto cadena que deteriora varios elementos cada vez que se produce un fallo en un motor.

8.1.2. Estructura del robot

Los experimentos han demostrado que la versión de *cadera estrecha* ha dado mejores resultados que la otra versión. La *versión ancha* parece más estable, haciendo que sea más complicado balancear el robot. Se ha visto que este movimiento es fundamental para poder hacerle caminar, por lo que interesa orientar el diseño del robot a la estabilidad sin excederse ya que puede ser complicado poder moverlo.

Una desventaja de este diseño es que no existen puntos de corte en ninguno de los ejes de las piernas. Al tener ejes que se cruzan, en lugar de cortarse, se complica mucho la cinemática del robot, al no existir articulaciones esféricas. Si se cortasen los dos ejes del tobillo, se podrían utilizar técnicas, como la de *desacoplo cinemático*, que permitirán poder realizar un análisis cinemático más exhaustivo.

Por otro lado, existen movimientos que provocan colisiones entre diferentes elementos estructurales o entre motores. En el primer caso, existen colisiones entre los eslabones l_3 y l_4, así como en l_6 y l_7. Este diseño provoca que se limite los recorridos de esos ejes, en función del movimiento del otro elemento de la pareja. Esto es necesario tenerlo en cuenta si no se quieren forzar los motores al bloquear su movimiento. Este problema se podría solucionar rediseñando los puntos de sujeción y la forma de estos. Además, volviendo al párrafo anterior, estos pares de ejes no se cortan en un punto. Si lo hicieran las combinaciones en los que se produce una colisión se disminuirían.

Existen casos en las que las colisiones se originan al entrar en contacto las cubiertas de los servomotores, concrétamente en los motores de la rodilla al chocar con los motores de la ingle o del tobillo. Estas colisiones solo se producen si no se mueve la rodilla de la forma que el

diseño a tenido en cuenta. Esto significa que los servos de la rodilla han de ser paralelos en todo momento al suelo. Afortunadamente, estas colisiones solo se producen en movimientos amplios, que no han sido utilizados para en este trabajo. No obstante, con una mejora de potencia y par en los motores, se podría sustituir la articulación doble de la rodilla por un solo eje. Esto evitaría en gran parte el problema de las colisiones de las carcasas de los servos, simplificando el diseño final del robot. Esta cambio además permitiría tener seis grados de libertad por pierna, pudiendo utilizar métodos más complejos para hallar la cinemática inversa del robot sin tener que aplicar restricciones añadidas.

Por lo demás, se ha visto que la estructura resiste perfectamente todos los esfuerzos que aparecen en el robot. Las piezas impresas[1] han soportado todos los esfuerzos sin deteriorarse y deformarse, ni si quiera en las uniones atornilladas entre las palancas de los servos y la pieza, que son los puntos donde se acumulan más tensiones.

8.1.3. Electrónica

Durante todos los experimentos se ha testado la electrónica del robot. En ellos se ha visto que el punto débil del sistema eléctrico es el regulador de tensión. El *UBEC* es el encargado de regular la tensión a bordo del robot, pasando a través de él toda la corriente demandada por la placa, por los motores y por el resto de dispositivos que se pretendan alimentar. Como se ha dicho en este mismo capítulo, los servomotores, al estar en estados transitorios permanentemente, demanda más corriente que la que el regulador puede ofrecer sin dañarse. Este elemento está diseñado para poder regular hasta 8 amperios y soportar breves periodos en los que puede soportar hasta 15. Se ha medido que la corriente que demanda el robot cuando ejecuta el movimiento de caminata optimizado ronda los 8 amperios, creando fluctuaciones que superan este valor. Esto provoca que el regulador trabaje a su máxima capacidad, y en ocasiones superándola brevemente. Esta intensidad se dispara en el caso de que varios de los motores ofrezcan su máximo par o que este coaccionados, obligando a pasar más corriente por el regulador. Estos sobreesfuerzos han provocado la rotura del regulador en dos ocasiones durante todo el proyecto cuando algún motor se ha roto.

La batería parece la más adecuada para el robot, al provocar tandas, gracias a la eficiencia del regulador de tensión, de una duración aproximada de entre 12 y 15 minutos, alternando todo tipo de movimientos. Este elemento no ha dado ningún problema durante el proyecto, ni en la carga ni descarga de este elemento. Al actuar el regulador de tensión conmutado, el robot no demanda tasas de descarga superiores a las que se permite.

Por otro lado, la placa de expansión que se utiliza para poder conectar los servomotores a la placa ha dado muy buen resultado. El inconveniente principal de esta placa es el problema de diseño que presenta, que ha sido explicado anteriormente. Este fallo grave hace que se cortocircuite la tensión de entrada con la masa del robot. Para evitarlo es necesario un separador que aísle ambas pistas, que hace que no encaje por completo el escudo dentro de la controladora. En ocasiones, al pasar tanta corriente por ella y tener unas pistas eléctricas relativamente pequeñas, se produce un leve calentamiento que es disipado fácilmente por los conectores de los servos, que actúan como aletas térmicas. Este calentamiento se produce en las pistas más cercanas a la alimentación de este escudo. Para evitar esto se puede dividir la alimentación entre dos conectores y disminuir la corriente que pasa por ellos.

8.1.4. Controladora

La placa ha mostrado grandes resultados, cumpliendo el objetivo principal de mover todos los motores del robot a tiempo real y de forma controlada. Se ha comprobado, sin detallarse en la fase de experimentos, que es capaz de tomar datos de sensores y procesarlos mientras ejecuta

[1] Se ha probado en los dos materiales más comunes que se emplean en impresoras 3D domésticas, que son ABS y PLA

tareas de movimiento. Esto, además que para el proyecto, confirma que la placa es apta para el robot, pudiendo competir en algunas pruebas del concurso *CEABOT*.

La placa ha sido alimentada directamente con el regulador de tensión[2] por las patillas `GND` y **Vcc** de la placa de expansión que, a su vez alimentan la placa por `GND` y **5V** directamente, sin usar el regulador de tensión. Pese a que trabaja a mayor tensión, la placa ha funcionado correctamente durante todo el proyecto, solo sufriendo un pequeño calentamiento sin ninguna consecuencia.

Esta placa presenta como desventaja una extensión elevada para la cantidad de pins y recursos que presenta. Realmente no se utiliza la totalidad de los pines, quedando mucho de ellos libres y sin uso. Tampoco se utiliza ningún convertidor analógico al no existir la necesidad de leer alguna señal analógica de baja frecuencia. No obstante, al usar 24 motores, se utilizan 4 de los 6 *TAUs* de la placa, ya que se deben usar funciones que contabiliza el tiempo y la generación de todas las señales *PWM* para el control de los motores. En el caso de usar un motor más, por ejemplo para una articulación extra en el cuello para que puede mover la cabeza de arriba a abajo y a los lados(*pan-tilt*), se tendría que habilitar un *timmer* adicional según muestra el creador de la librería.

8.1.5. Cálculos previos

Se ha visto que los movimientos se corresponden con los cálculos hecho. Todos los cálculos realizados para cada *submovimiento* mueven al robot la distancia indicada, con un error muy bajo. Este error se debe más a la falta de exactitud de los motores ocasionada por la holgura de estos. Pese a no se un método utilizado en la robótica humanoide, produce unos resultados suficientemente exactos que se pueden considerar como válidos.

El mayor inconveniente que presenta este método es que, aunque los estudios movimientos sean simples y solo dependan de un único parámetro cada uno de ellos, las trayectorias que se utilizan para moverlos requiere más esfuerzo. Sobre todo, es difícil ajustar cada uno de los parámetros para conseguir un movimiento válido.

Desgraciadamente no se incluyen cálculos dinámicos dentro de estos cálculos, teniendo que realizar un gran número de experimentos para poder conseguir resultados positivos. Por otro lado, estos cálculos podrían servir para descartar previamente alguna versión o movimiento sin tener que recurrir a experimentos.

8.1.6. Programas diseñados

Los programas diseñados han dado buenos resultados, tanto para este proyecto como para un uso más general.

En primer lugar, la librería `MYOD` se ha probado en diferentes robots que usan una controladora *Arduino* y motores que se controlan a través de una señal *PWM*. Todas las funciones que están definidas funcionan correctamente, facilitando la programación al poder usar todas ellas en un objeto de programación. No obstante, no se ha probado esta librería junto a una interrupción[3], por lo que no se puede demostrar empíricamente el buen funcionamiento de esta librería al usarla con interrupciones que puedan interferir con la escala de tiempos.

El programa de *Trim* funciona correctamente, ya que cumple su cometido al almacenar y modificar los valores de ajuste de cada motor. Estos valores se han leído correctamente en otros programas que necesitan usar estos datos, comprobando la utilidad y validez de este programa. En contraposición, este programa guarda los valores de forma no consecutiva al dividir los datos en dos celdas de memoria. Esto puede ser un inconveniente si se quiere modificar el número de

[2] El voltaje medido a la salida del regulador es de 5,8 voltios cuando está seleccionada la posición de `6V`

[3] *Arduino* dispone de diferente librerías que permiten usar interrupciones al apoyarse en alguno de los *timmers* disponibles. Ejemplos de estas librerías sin `TimmerOne`, `TimmerThree` y `TimmerFive`.

motores que se utilizan, al desplazar todos los valores una celda. Por lo tanto, sería necesario borrar de nuevo la memoria para no provocar una lectura errónea de los datos. Por otro lado, este programa se ha empleado también para inclinar el robot para ajustar el movimiento de caminata.

También se ha podido ver que el conjunto de programas que generan caminatas han sido válidos y eficaces para este proyecto. Estos programas, que componen el núcleo de este trabajo, se han tenido que ir modificando para adaptarlo a las nuevas condiciones que han ido surgiendo a lo largo del proyecto, en concreto para ajustar la inclinación y deriva del robot. Los valores umbral que permiten ajustar las trayectorias están incrustados en el programa, siendo parámetros que no se pueden cambiar durante la ejecución del programa. Para variarlos, es necesario volver a compilar el programa incluyendo la modificación. Por otro lado, ocurre lo mismo con el centro de gravedad, que está definido del mismo modo.

Este programa además permite cargar configuraciones previas. En esta configuración se guarda la identidad de cada motor y saber que función ocupa centro del vector. En ocasiones, se puede hacer repetitivo la introducción de los datos para generar las trayectorias. Para evitar introducir un dato adicional, se puede guardar la longitud de la pierna dentro de la configuración del programa.

Se han obtenido excelentes resultados en el programa *Generador de movimientos por cinemática directa*. Este programa ha permitido generar de manera sencilla y rápida movimientos destinados a diferentes objetivos, entre ellos la generación de los movimientos de giro y de desplazamiento lateral. Los principales inconvenientes de este programa son que no guarda los datos en un fichero, teniendo que recurrir al *Conversor de texto a formato de código*, y que no se dispone de un entorno gráfico que simplifique algunas tareas. El primer inconveniente es un problema de fácil solución, pero que puede hacer, si el usuario no se acuerda de guardar la información que se muestra por pantalla, que se pierda todo el trabajo realizado hasta el momento si se desconecta el cable o se apaga la placa. El segundo problema es más complicado de solucionar, al comunicar la placa mediante una comunicación *serie* que sólo muestra los datos por pantalla. Esto hace que la interfaz sea poco atractiva para el usuario, pareciendo poco entendible y manejable.

Por último, el *Conversor de texto a formato de código* cumple perfectamente la función para la que ha sido diseñado, al leer los ficheros de texto en los que se han guardado los movimientos y traducirlos a un fichero nuevo que el *software* de *Arduino* es capaz de leer perfectamente. Este programa se adapta al número de movimientos movimientos y motores que estén indicados en el archivo.

Cabe destacar que todos los programas diseñados para este proyecto adaptan a cada problema que se pretenda diseñar con estos métodos al número de elementos actuadores que tenga el robot. Esto permite orientar la librería `MYOD` a cualquier tipo de robot con este tipo de motores y controladora, sin importar la configuración estructural que tenga. Para adaptar este cambio, la librería disponen de un valor, llamado `Nmotor`, que se puede cambiar que hace referencia al número máximo que la librería puede controlar, siempre dentro de las limitaciones de la placa *Arduino* que se utilice.

8.1.7. Experimentos

Los experimentos vistos en el tema anterior han demostrado la viabilidad de ciertos movimientos y han denotado que la versión de *cadera estrecha* permite más movilidad que la otra versión. En contra partida, estos experimentos han sido laboriosos y de larga duración en el tiempo, al tener que probar múltiples combinaciones de variables. La optimización del movimiento de caminata es un procedimiento iterativo, debiendo realizar un número alto de veces los experimentos de ajuste de umbrales y de amplitudes.

No obstante, los experimentos han servido para obtener las diferentes conclusiones vistas hasta ahora.

8.1.8. Conjunto del proyecto

El conjunto se puede considerar como exitoso, al haber cumplido todas las expectativas descritas en el apartado 1.2.

- Se ha diseñado un sistema eléctrico que cumple todas las funciones básicas que se necesitan para el proyecto, dando opción a posibles expansiones en cuento a sensores o nuevos actuadores que se puedan incluir en un futuro.
- Se ha conseguido definir diferentes programas que permiten la total movilidad y funcionamiento del robot. Además, la librería de este proyecto se puede reutilizar para otros proyectos en los que se utilicen elementos similares.
- Este proyecto ha servido para encontrar fallos y problemas del diseño mecánico del robot, dejando anotaciones de cómo poder subsanarlos en un futuro en el caso se que este proyecto siga adelante.
- Todos los dados experimentales se pueden exportar a otros robots del mismo modelo siempre que se sigan los pasos detallados a lo largo del proyecto, evitando tener diferentes programas con distintos valores para realizar una misma función.
- Y, por supuesto, se ha conseguido hacer mover el robot por el plano. El movimiento de éste está formado por el movimiento de caminata en línea recta, que se ha optimizado para este robot, además de los diferentes movimientos que le permiten girar y levantarse.
- Basándose en el punto anterior, se pudo presentar el prototipo a la competición *CEABOT 2014*, donde se pudo participar en las pruebas de combate y obstáculos. Por lo tanto, se consiguió otro objetivo final al conseguir participar en la competición para la que fue diseñado contra competidores que superaban varias veces el precio total del robot.

Por otro lado, se ha visto que puede ser casi imprescindible un movimiento que permita levantase al robot tras caerse al suelo. Este movimiento necesita una gran fuerza en los brazos, y por lo que se ha visto experimentalmente en este proyecto, unos brazos de mayor longitud que los actuales[4], que facilitarían enormemente esta labor.

Se ha observado también que los motores permiten la movilidad del motor, aunque trabajan en ocasiones al límite de sus capacidades. Esto ha hecho que el proyecto presente numerosos inconvenientes que no se producirían con motores de mayor calidad y fuerza, al disminuir las holguras y aumentar la fuerza, la potencia y el control de estos.

8.2. Posibles mejoras del diseño

Basándose en la experiencia adquirida a lo largo del proyecto, se muestran a continuación un conjunto de posibles modificaciones a realizar para mejorar el diseño.

En primer lugar, se recomienda el cambio de los motores a unos que presenten mejores prestaciones. Dentro de éstas prestaciones sería recomendable incluir mejores características de torque, sin importar tanto la velocidad de movimiento. Esto significa que se pueden elegir motores de igual potencia, que hace que no aumente el consumo de forma significativa. Por otro lado, sería importante saber la posición real de los motores, o al menos la que lee el encoder,

[4]Este diseño no está totalmente acabado, al no estar definido el último o últimos eslabones de los brazos.

evitando hacer suposiciones de la posición final en la que se encuentra realmente la articulación. Sería interesante disponer de un motor que mostrase el esfuerzo de par que produce de forma real el motor. Esto permitiría poder usar algún algoritmo, como el *ZMP*, que permite integrar un lazo de control de los que se usan el robot de mayor escala y complejidad. Por último, convendría que el motor tuviera su propio microcontrolador integrado que permita manejar el motor a través de él y poder leer y escribir en algunos de sus registro para facilitar parte de la tarea a la controladora.

Se ha visto con relativa frecuencia que existen colisiones entre diferentes motores o partes estructurales. En relación con esto, tampoco se producen articulaciones esféricas en ninguna parte de la pierna. Por esto, se propone que la nueva estructura tenga en cuenta en el diseño el corte de diferentes ejes, como el grupo de q_1, q_2 y q_3 y la pareja q_6 y q_7. Esto sería posible si se recolocasen los motores en distribuciones similares a las estructuras de los robots *Bioloid* y *Kondo*, visto en 3.2.

También se ha observado que para balancear al robot no se necesita una cadera tan ancha, siendo más fácil crear aceleraciones al acortar esta distancia, sin tener que recurrir a variar la longitud de la pierna. Por esto se recomienda juntar algo más las piernas, siempre que la holgura de los motores lo permita.

Por otro lado, sería recomendable, al usar motores con mayor par, poder quitar uno de los dos motores que componen la rodilla. Esto tendría notables ventajas, como disponer de un brazo robótico de seis grados de libertad y un descenso en las colisiones de las carcasas de los motores. Además, rebajaría el peso total del robot y de los esfuerzos que tendrían que vencer el resto de motores al tener que desplazar menos masa e inercia. Por último, a causa de lo anterior y de disponer de un motor menor, el consumo energético del robot disminuiría.

En cuanto al sistema eléctrico, viendo el estrés que sufre el regulador de tensión, sería conveniente usar otro regulador conmutado que pueda trabajar con una corriente nominal superior. Se ha visto que este regulador debe tener al menos una corriente nominal de al menos 10 *amperios*, soportando picos superiores. Otra opción sería dividir la carga de trabajo que soporta el regulador, dividiendo el circuito en partes más pequeñas. Por ejemplo, se podrían usar dos reguladores diferentes que alimenten las piernas con uno y el resto con el otro. Esto implicaría modificar la placa de expansión, al no poder ser alimentada al provocar un cortocircuito virtual que generaría una corriente enorme desde un regulador hacia el otro. Por lo tanto, para usar este sistema, sería necesario diseñar una nueva placa con dos circuitos independientes que tengan las tierras comunes, con una forma similar al escudo actual.

8.3. Desarrollos futuros

En este proyecto se ha optado por usar el método trigonométro para obtener las cinemáticas inversas y poder mover así el robot. El inconveniente del método de superposición de movimientos es que orienta el estudio al movimiento final que se pretende calcular, teniendo que realizar otro análisis diferente considerando otras restricciones para hallar otro movimiento. Por eso se pretender calcular de nuevo la cinemática usando el método que se ha descartado.

Si se utilizase el método matricial, acompañado de un rediseño del robot según lo que se muestra antes, se podría calcular sin muchas dificultades la cinemática inversa por ese método. La matriz jacobiana tendría unas dimensiones [6x6], si se quitan un motor de la rodilla, siendo el espacio solución igual al número de variables. Esto permite invertirla fácilmente al ser una matriz cuadrada, obteniendo la cinemática inversa de un modo similar al utilizado en brazos robóticos y robots humanoides.

El robot no dispone de ningún sensor inercial, no pudiendo crear un lazo de control que permite al robot tener equilibro. Para obtener un valor del ángulo en el que se encuentra el

robot respecto a la vertical. Para conseguir esto, es necesario utilizar un sensor giroscópico. Este sensor no da inclinaciones, si no velocidades angulares que se referencia a cada uno de los tres ejes. Par obtener la inclinación sería necesario integrar cada uno de estos valores. Al integrar se produciría un pequeño error que se iría acumulando con el tiempo, dando un valor erróneo. Para evitar esto, pude ser necesario incluir un *filtro de Kalman* que se apoye en una brújula electrónica para disminuir el error en el eje vertical, sirviendo de referencia para los otros dos.

Además, se podría incluir poder usar el método de *Zero Moment Point* si se utilizasen motores que indicasen el consumo de energía o, más concrétamente, el par ejercido en ese instante. Junto a una estructura que hiciera que se cortasen los dos ejes del tobillo en un solo punto perpendicularmente, se podría leer estos valores y considerarlos como la descomposición de las reacciones que se producen en dicha articulación. Estos valores, junto con una cinemática inversa correcta como la del método matricial, se puede crear un lazo de control que permite generar caminatas y movimientos con correcciones a tiempo real.

Para poder realizar estos cálculos a bordo del robot y que siga siendo autónomo, es necesario una controladora con mayor velocidad de procesamiento de datos. Se ha visto que la *Beaggle Bound Black* puede ser capaz de generar las señales de control de los motores gracias a sus *timmers*. Esta placa tiene una velocidad de procesamiento muy elevada si se compara con la controladora actual. Dentro de esta *SBC* se pueden realizar todos los cálculos a tiempo real de lo que se ha hablado hasta ahora. Por lo tanto, se podría orientar un nuevo proyecto a lo visto hasta ahora.

Por otro lado, es necesario rediseñar el circuito eléctrico, en concreto la parte relativa a la regulación de tensión. Como se ha explicado antes, es necesario crear un nuevo diseño para la placa de expansión, tanto si se sigue utilizando la controladora actual como si se implanta la placa anterior. En el primer caso sería necesario este nuevo diseño ya que se debería crear dos circuitos independientes para alimentar dos regiones diferentes del robot, estando alimentadas cada una por un regulador de tensión diferentes. La placa actual no se amolda al nuevo requisito, por lo que este cambio provoca que haya que diseñar una placa sencilla al no existir una solución comercial alternativa. En el otro caso, el relativo a cambiar la controladora, es necesario crear el *shield* que incluya la característica anterior ya que no existe ninguna placa en el mercado que se ciña a esto. Por otro lado, se podrían usar dos reguladores comerciales de la misma gama en lugar de tener que diseñarlo, aunque es una opción que queda abierta.

Para facilitar la programación del robot, se pueden crear entornos gráficos para los programas diseñados. Esto puede incluir también un modelado del robot, que permita poder probar nuevos movimientos de modo virtual sin tener que tener que poner el robot en marcha para probarlo. Por ello, puede ser necesario crear un modelado del robot en plataformas como *OpenRave*, *V-Rep Pro*, *WeBots* o cualquier otro programa que permita el control del robot en un entorno virtual.

Capítulo 9

Gestión del proyecto

Por último, en este capítulo se muestran los costes parciales de cada elemento del prototipo así como el coste total del proyecto[1][2].

9.1. Estructura y motores

9.1.1. Coste de los servomotores para versión mixta

	Precio	Cantidad	Total
TowerProMG996R	8.50	15	127.5 €
Futaba S3003	8	8	64 €
TowerProMG90S	4		195.5 €
			195.50 €

9.1.2. Coste de los servomotores para la versión TowerPro

	Precio	Cantidad	Total
TowerProMG996R	8.50	23	195.50 €
TowerProMG90S	4.00		4.00 €
			199.50 €

9.1.3. Coste de la estructura

	Precio	Cantidad	Total
Plástico ABS/PLA	20.00€/ Kg	∼450g	9.00 €
Tornillería			6.00 €
Consumo Energético			∼ 3.00 €
			18.00 €

[1]Los precios de los elementos son aproximados ya que existen fluctuaciones en el valor de estos en función del estado del mercado

[2]Los precios incluyen impuestos indirectos

9.1.4. Coste total de la estructura y motores

	Precio	Cantidad	Total
Motores			199.50 €
Estructura			18.00 €
			221.50 €

9.2. Electrónica

	Precio	Cantidad	Total
Arduino Mega 2560			∼ 15.00 €
UBEC 8A			12.00 €
Batería			8.50 €
Shield			12.00 €
Conector XT60	0.90	1	0.90 €
Extensores Cable Servo	0.20	4	0.80 €
			49.20 €

9.3. Costes de mano de obra

	Precio	Cantidad(h)	Total
Investigación	25 €/h	80	2000 €
Cálculos teóricos		80	2000 €
Diseño Librería MYOD		50	1250 €
Diseño sistema eléctrico		40	1000 €
Diseño prog Trim		8	200 €
Diseño prog Generador		80	2000 €
Diseño prog Cinem. Directa		10	250 €
Diseño prog Conversor		20	500 €
Pruebas experimentales		180	4500 €
Redacción		150	3750 €
		698	17450 €

9.4. Coste Total

	Total
Estructura y motores	221.50 €
Electrónica	49.20 €
Mano de obra	17450.00 €
	17720.70 €

Bibliografía

[1] Arduino. Arduino mega 2560. [Disponible online Septimebre 2014] http://arduino.cc/en/Main/ArduinoBoardMega2560.

[2] Arduino. Intel galileo. [Disponible online Septimebre 2014] http://arduino.cc/en/ArduinoCertified/IntelGalileo.

[3] CEABOT. Normativa de viii concurso de robots humanoides, [Disponible online en Septiembre 2014]. www.ceautomatica.es/sites/default/files/upload/10/files/normativa%20CEABOT_2014.pdf.

[4] Kondo Kangaku co. [Disponible online Septimebre 2014]http://kondo-robot.com/product/03071.

[5] DARPA. The robot challenger, [Disponible online en Septiembre 2014]. www.theroboticschallenge.org.

[6] Comité Español de Automática. Competición ceabot, [Disponible online en Septiembre 2014]. www.ceautomatica.es/og/robotica/concurso-ceabot.

[7] Grupo de MiniHumanoides de ASROB. Proyecto myod, [Disponible online en Septiembre 2014]. http://ieee.uc3m.es/index.php/MYOD.

[8] Asociación de Robótica ASROB. Pagina web ofcicial, [Disponible online en Septiempre 2014]. http://ieee.uc3m.es/index.php/Main_Page.

[9] Raspberry Pi Foundation. Raspberry pi. [Disponible online Septimebre 2014] www.raspberrypi.org/.

[10] The BeagleBoard.org Foundation.

[11] M. González-Fierro, A. Jardón, S. Martínez de la Casa, M.F. Stoelen, J.G. Víctores, and C. Balaguer. Educational initiatives related with the ceabot contest.

[12] HobbyKing, [Disponible online en Septiembre 2014]. www.hobbyking.com.

[13] Intel. [Disponible online Septimebre 2014] www.intel.es.

[14] B Borovac M. Vukobratovic and V. Potkonjak. Zmp: A review of some basic misunderstandings. 2006.

[15] D. Surla M. Vukobratovic, B. Borovac and D. Stokic. Biped locomotion: Dynamics, stability, control and application. 1989.

[16] A.Barrientos L.F.Peñín C.Balaguer R.Aracil. *Fundamentos de robótica*. McGraw Hill, 1997.

[17] Ro-Botica. [Disponible online Septimebre 2014] http://ro-botica.com/es/Producto/Actuador-Dynamixel-AX-12A/.

[18] RoBoard. Roboard rb-110. [Disponible online Septimebre 2014] http://www.roboard.com/RB-110.htm.

[19] Robotis. Do-it-yourself educational robot kit. [Disponible online Septimebre 2014] www.robotis.com/xe/bioloid_en.

[20] ServoDatabase, [Disponible online en Septiembre 2014]. www.servodatabase.com.

[21] SuperRobótica. Robonova 1 el robot humanoide para todos los públicos., 2014. [Disponible online Septiembre 2014] http://www.superrobotica.com/robonova.htm.

[22] M. Vukobratovic and B. Borovac. Zero-moment point - thirty five years of its life. 2004.

[23] J.Denavit y R.S.Hartenberg. A kinematic notation for lower-pair mechanism based on matrices. *Journal of Applied Mechanics*, 1995.